Fraction-free Method for Computing Rational Normal Forms of Polynomial Matrices

SHUICHI MORITSUGU[†] AND KAZUKO KURIYAMA[‡]

[†] *University of Library and Information Science*

moritsug@ulis.ac.jp

[‡] *University of Tsukuba*

kkuriyam@iris.is.tsukuba.ac.jp

(Received 11 June 1997)

In this paper, we present a fraction-free algorithm for computing rational normal forms of matrices with univariate polynomial entries. In the traditional algorithm based on Danilevskii's method, its principal transformations are similar to Gaussian elimination. When they are carried out exactly by computer algebra system, the elements of intermediate matrices extremely swell and it requires much CPU-time to compute g.c.d.s of polynomials for reducing rational expressions.

In order to avoid such difficulty, we give a fraction-free algorithm analogous to Bareiss' single-step fraction-free elimination. We have implemented it on the computer algebra system Reduce3.5. The experimental result shows the efficiency of our algorithm.

1. Introduction

In this paper, we present a fraction-free scheme in similar transformations of matrices. Using this scheme, a fraction-free algorithm for computing rational normal forms (or Frobenius normal forms) of square matrices is given. Experimental result on the matrices, whose elements are univariate polynomials with integer coefficients, shows the efficiency of the algorithm. The essential parts of this paper, i.e., algorithms and result of experiment, are the tranlation from the authors' already published paper (Kuriyama and Moritsugu, 1996).

Rational normal forms (Gantmacher, 1959) are important for the symbolic computation of Jordan normal forms of integer matrices (Gil, 1992) (Moritsugu and Kuriyama, 1992)(Kuriyama and Moritsugu, 1994). In these papers, it is reported that the CPU-time required for computing rational normal forms is dominant in the whole CPU-time. In order to extend it to the computation of polynomial matrices, the algorithm for computing rational normal forms has to be improved on its efficiency, because it requires much more CPU-time than that for integer matrices.

Kaltofen *et al.* (1990) shows parallel algorithms for the Smith normal forms of a matrix with univariate polynomial entries. The rational normal forms of a matrix A can be constructed through the Smith Normal Form of its characteristic matrix $\lambda E - A$. Consequently, they propose efficient algorithms for rational normal forms and Jordan normal forms. However, our approach is the direct transformation of matrix A into its rational normal form by a similar transformation $A \mapsto S^{-1}AS$.

On the other hand, fraction-free scheme has been used as an important idea for the improvement of efficiency of algorithms such as:

Reduced polynomial remainder sequence (Collins, 1966)(Collins, 1967),

Fraction-free Gaussian elimination (Bareiss, 1968).

Sasaki and Murao (1982) proposed a modification of fraction-free Gaussian elimination and showed efficient algorithms for matrices with multivariate polynomial entries. Rouillier (1995) extends Bareiss' method to the diagonalization of symmetric matrices by a congruent transformation $A \mapsto S^tAS$, where A is symmetric. The result is represented by similar notations to Bareiss' formula. Our algorithm for rational normal forms $A \mapsto S^{-1}AS$, where A is not necessarily symmetric, is based on Danilevskii's method(Danilevskii, 1937). The procedure is represented by composition of elementary transformations for matrices(Kan and Iri, 1982), which are analogous to Gaussian elimination.

Although Danilevskii's method has the best computational complexity of all algorithms for computing characteristic polynomials of matrices, it is not practical for numerical computation because of the problem of numerical accuracy. As for exact computation by computer algebra with infinite precision, it is no matter and Danilevskii's method is very useful(Takeshima *et al.*, 1991). However, in the symbolic computation, the elements of the intermediate matrices are almost always swelling rational expressions, even though the elements of input matrix are integers or polynomials, because the operations for transformations include division. Nevertheless, for example, when the elements of input matrix are integers, the output rational normal form is also a integer matrix. Therefore, the intermediate rational expressions are sure to cancel finally.

We discovered that the cancellation rule of common denominators holds in the process of similar transformations of matrices (Kuriyama and Moritsugu, 1996), which is analogous to Bareiss' Single-step fraction-free elimination. We have implemented it on the computer algebra system Reduce3.5 and experimented on various sizes of matrices, with the result that the CPU-time is reduced by around 70% at most from the traditional algorithm.

2. Method for computing Rational Normal Forms

2.1. Basic notions

In the following chapters, we assume that the elements of a given matrix are univariate polynomials with integer coefficients, that is, $A = [a_{ij}]$, $a_{ij} \in \mathbf{Z}[x]$.

Definition 2.1. (Companion matrix) *The following $q \times q$ square matrix*

$$A = \begin{bmatrix} 0 & 0 & \cdots & 0 & a_0 \\ 1 & 0 & \cdots & 0 & a_1 \\ & \ddots & \ddots & \vdots & \vdots \\ & & \ddots & 0 & a_{q-2} \\ & O & & 1 & a_{q-1} \end{bmatrix} \tag{2.1}$$

is called the companion matrix associated with the polynomial $f(\lambda) = \lambda^q - a_{q-1}\lambda^{q-1} - \cdots - a_1\lambda - a_0$. In particular, the companion matrix associated with the first degree polynomial $f(\lambda) = \lambda - a_0$ is the 1×1 square matrix $[a_0]$.

The characteristic polynomial $\varphi_A(\lambda)$ and the minimal polynomial $\phi_A(\lambda)$ of the companion matrix A are equal to $f(\lambda)$.

Theorem 2.1. (Rational Normal Form) *Using a suitable regular matrix S, every $n \times n$ square matrix A can be transformed into a block diagonal matrix as follows*:

$$S^{-1}AS = C_1 \oplus C_2 \oplus \cdots \oplus C_t. \tag{2.2}$$

It is called the rational normal form (or Frobenius Normal Form) of A. Each block matrix $C_i (i = 1, \cdots, t)$ is a $q_i \times q_i$ companion matrix (2.1), *and the minimal polynomial $\phi_{C_{j+1}}(\lambda)$ of C_{j+1} divides the minimal polynomial $\phi_{C_j}(\lambda)$ of C_j $(j = 1, \ldots, t-1)$. The matrix in the form* (2.2) *always exists and is uniquely determined for every given matrix.*

Theorem 2.2. (Rational Normal Forms and elementary divisors) *Let the elementary divisors except 1 in descending order of their degrees be $e_1(\lambda), \ldots, e_p(\lambda)$, which are contained in the Smith Normal Form of the characteristic matrix $\lambda E - A$. If the rational normal form R of A has the following form,*

$$R = C_1 \oplus C_2 \oplus \cdots \oplus C_p \tag{2.3}$$

each block matrix C_i $(i = 1, \ldots, p)$ is the companion matrix associated with the elementary divisor $e_i(\lambda)$, and the minimal polynomial $\phi_{C_i}(\lambda)$ coincides with $e_i(\lambda)$ for each i.

In order to compute a similar transformation $A \mapsto S^{-1}AS$, we apply the following transformations successively.

Definition 2.2. (Elementary transformations) *The following three operations are called the elementary transformations.*

- $op1(k, l)$: *Exchange the k-th row and the l-th row of A, and then exchange the k-th column and the l-th column of A.*
- $op2(k, c)$: *Multiply the k-th row of A by c^{-1}, and then multiply the k-th column A by c.*
- $op3(k, l, c)$: *Add the l-th row multiplied by c to the k-th row of A, and then subtract the k-th column multiplied by c from the l-th column of A.*

Since the operations to rows correspond to the transformation matrix S^{-1}, the operations to columns correspond to S, and $(S^{-1}A)S = S^{-1}(AS)$, it gives the same result if we apply the operations to columns first in definition 2.2. As for the above operations, it is no matter which operation to the row or to the column is carried out first and another is second. When A is transformed into $\tilde{A}$ by the above elementary transformations, $\tilde{A}$ is similar to A and they have the same rational normal form in common.

2.2. THE STRAIGHTFORWARD ALGORITHM

Let $A = [a_{i,j}]\,(i, j = 1, \ldots, n)$ be a $n \times n$ square matrix. In the traditional algorithm, the following operations are carried out repeatedly as its main step for each companion block. We suppose that $\tilde{A} = [\tilde{a}_{i,j}]$ denotes the A transformed by the elementary transformations independently of the times of operations.

Algorithm 1 (Straightforward similar transformation)

```
( p ← 0 ;  % p is the number of the companion blocks.
  k ← 1 ;  % k is the index of the pivot element ã_{k+1,k}.
  % These initial values are set up once at the beginning of the computation. )
p ← p + 1 ;
kk ← k ;   % kk is the index of the principal element of C_p in Ã.
while k < n do
   { l ← k + 1 ;
     while (l ≤ n) & (ã_{l,k} = 0) do l ← l + 1 ;
     if l > n then break ;
              else { op1(k + 1, l) ;
                     op2(k + 1, ã_{k+1,k}) ;
                     for i ← 1 to k do op3(i, k + 1, −ã_{i,k}) ;
                     for i ← k + 2 to n do op3(i, k + 1, −ã_{i,k}) ;
                     k ← k + 1 }} ;
m_p ← kk − k + 1 ;   % m_p is the size of the pth companion block.
```

We can compute the transformation matrices S, S^{-1} simultaneously by applying the above algorithm to identity matrices. Separating the operations contained in the algorithm for rows and for columns, we apply the former to an identity matrix E_1 and the latter to another identity matrix E_2. Then, we get $(E_1 \to \cdots \to S^{-1},\; E_2 \to \cdots \to S)$, where $S^{-1}AS$ is the rational normal form of A.

Showing the case where $n = 4$ and the only one companion block is contained, we illustrate how the above algorithm proceeds.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \to \begin{bmatrix} 0 & \star & \star & \star \\ 1 & \star & \star & \star \\ 0 & \star & \star & \star \\ 0 & \star & \star & \star \end{bmatrix}$$

$$\to \begin{bmatrix} 0 & 0 & \star & \star \\ 1 & 0 & \star & \star \\ 0 & 1 & \star & \star \\ 0 & 0 & \star & \star \end{bmatrix} \to \begin{bmatrix} 0 & 0 & 0 & \triangle \\ 1 & 0 & 0 & \triangle \\ 0 & 1 & 0 & \triangle \\ 0 & 0 & 1 & \triangle \end{bmatrix} \tag{2.4}$$

Note that $\star$s are rational expressions of a_{ij} but $\triangle$s are polynomials of a_{ij}. Since the elementary transformations include divisions, the elements of the intermediate matrices are rational expressions in general. However, the elements in the rightest column of the companion matrix are equal to the coefficients $-c_i$ $(i = 0, \ldots, n-1)$ of A's characteristic polynomial $\varphi_A(\lambda) = \det(\lambda E - A)$, and they are always polynomials of a_{ij} because of the rule for the computation of $\det(\lambda E - A)$.

In the above process, extra CPU-time is needed for the reduction of rational expressions by computing g.c.d.s. Moreover, the intermediate results require much more memory because the common denominators swell as the computation proceeds.

3. Improvement on the efficiency of the algorithm

3.1. BASIC NOTIONS

The operations used for the elimination in **Algorithm 1** are similar to the ones used for Gaussian elimination in the matrix triangulation. In computing the upper triangular matrix by Gaussian elimination, we have only to apply row operations: $A \mapsto GA$. However, in computing the rational normal form, we have to apply both row and column operations: $A \mapsto S^{-1}AS$, in order to keep the similarity of matrices.

For the improvement on the efficiency of Gaussian elimination, Bareiss' fraction-free elimination algorithm is well-known. This algorithm is based on the fact that the common denominator of the rational expressions in the kth step divides all the elements after the $(k+1)$th step $(k = 1, \ldots, n-1)$. We attempted to apply similar device to the computation of rational normal forms, and we have discovered that the similar property of cancellation holds in the **Algorithm 1**. Here we show the principle of the fraction-free similar transformation.

Let $A = [a_{i,j}]\,(i, j = 1, \ldots, n)$ be a $n \times n$ square matrix. Starting from $A = A^{(0)} = \tilde{A}^{(0)}$, we suppose that $A^{(k)}$ denotes the matrix after the kth step and $\tilde{A}^{(k)}$ denotes the transformed $A^{(k)}$ by $op2(k+1, 1/\tilde{a}^{(k-2)}_{k,k-1})$. Let $\tilde{a}^{(k)}_{i,j}$ be the numerator of the (i, j) element in $\tilde{A}^{(k)}$. (When $k = 1$, we let $\tilde{a}^{(k-2)}_{k,k-1} = 1$.)

We suppose $a_{2,1} \neq 0$. (If $a_{2,1} = 0$, then try to find l $(l > 2)$ so that $a_{l,1} \neq 0$ and execute $op1(2, l)$.) For the first step elimination, we apply $op2(2, a_{2,1})$ to A and apply $op3(i, 2, -a_{i,1})$ $(i = 1, 3, \ldots, n)$. Then, by putting out the common denominator $a_{2,1}$, we get

$$\tilde{A}^{(1)} = \frac{1}{a_{2,1}} \begin{bmatrix} 0 & \tilde{a}^{(1)}_{1,2} & \tilde{a}^{(1)}_{1,3} & \cdots & \tilde{a}^{(1)}_{1,n} \\ a_{2,1} & \tilde{a}^{(1)}_{2,2} & a_{2,3} & \cdots & a_{2,n} \\ 0 & \tilde{a}^{(1)}_{3,2} & \tilde{a}^{(1)}_{3,3} & \cdots & \tilde{a}^{(1)}_{3,n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \tilde{a}^{(1)}_{n,2} & \tilde{a}^{(1)}_{n,3} & \cdots & \tilde{a}^{(1)}_{n,n} \end{bmatrix}. \tag{3.1}$$

In similar way, we suppose $\tilde{a}^{(1)}_{3,2} \neq 0$, and apply $op2(3, \tilde{a}^{(1)}_{3,2})$, $op3(i, 3, -\tilde{a}^{(1)}_{i,2})$ $(i = 1, 2, 4,$

$\cdots, n)$ to $\tilde{A}^{(1)}$, then put out the common denominator $\tilde{a}^{(1)}_{3,2}$, we get

$$A^{(2)} = \frac{1}{a_{2,1}\tilde{a}^{(1)}_{3,2}} \begin{bmatrix} 0 & 0 & a^2_{2,1}\triangle & a_{2,1}\triangle & \cdots & a_{2,1}\triangle \\ a_{2,1}\tilde{a}^{(1)}_{3,2} & 0 & a^2_{2,1}\triangle & a_{2,1}\triangle & \cdots & a_{2,1}\triangle \\ 0 & \tilde{a}^{(1)}_{3,2} & a_{2,1}\triangle & \triangle & \cdots & \triangle \\ 0 & 0 & a^2_{2,1}\triangle & a_{2,1}\triangle & \cdots & a_{2,1}\triangle \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & a^2_{2,1}\triangle & a_{2,1}\triangle & \cdots & a_{2,1}\triangle \end{bmatrix}, \tag{3.2}$$

where $\triangle$s are polynomials of $a_{i,j}$. Each element of the above matrix except for the third row contains $a_{2,1}$ as a common factor. (See also appendix.) Since each element of the third column except the $(3,3)$ element contains $a^2_{2,1}$ as a common factor, applying $op2(3, 1/a_{2,1})$ to $A^{(2)}$, we obtain

$$\begin{aligned} \tilde{A}^{(2)} &= \frac{1}{a_{2,1}\tilde{a}^{(1)}_{3,2}} \begin{bmatrix} 0 & 0 & a_{2,1}\triangle & a_{2,1}\triangle & \cdots & a_{2,1}\triangle \\ a_{2,1}\tilde{a}^{(1)}_{3,2} & 0 & a_{2,1}\triangle & a_{2,1}\triangle & \cdots & a_{2,1}\triangle \\ 0 & a_{2,1}\tilde{a}^{(1)}_{3,2} & a_{2,1}\triangle & a_{2,1}\triangle & \cdots & a_{2,1}\triangle \\ 0 & 0 & a_{2,1}\triangle & a_{2,1}\triangle & \cdots & a_{2,1}\triangle \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & a_{2,1}\triangle & a_{2,1}\triangle & \cdots & a_{2,1}\triangle \end{bmatrix} \\ &= \frac{1}{\tilde{a}^{(1)}_{3,2}} \begin{bmatrix} 0 & 0 & \triangle & \triangle & \cdots & \triangle \\ \tilde{a}^{(1)}_{3,2} & 0 & \triangle & \triangle & \cdots & \triangle \\ 0 & \tilde{a}^{(1)}_{3,2} & \triangle & \triangle & \cdots & \triangle \\ 0 & 0 & \triangle & \triangle & \cdots & \triangle \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \triangle & \triangle & \cdots & \triangle \end{bmatrix}. \end{aligned} \tag{3.3}$$

The former denominator $a_{2,1}$ is canceled and the common denominator of $\tilde{A}^{(2)}$'s elements turns out to be only $\tilde{a}^{(1)}_{3,2}$. Since **Algorithm 1** carries out the same procedure for each column successively, the common denominator in the kth step similarly divides all the elements after the $(k+1)$th step when $k \geq 3$.

Based on this property, the fraction-free elimination can be performed by multiplying whole $\tilde{A}^{(k-1)}$ by $\tilde{a}^{(k-1)}_{k+1,k}$ in advance. Since its common factor is canceled by transforming $A^{(k+1)}$ using $op2(k+2, 1/\tilde{a}^{(k-1)}_{k+1,k})$ after the kth elimination, the elements of the intermediate matrices do not swell.

3.2. FRACTION-FREE ELIMINATION ALGORITHM

The main step of **Algorithm 1** is improved as follows, based on the property shown in the previous section. Let $A = [a_{i,j}]\,(i,j = 1,\ldots,n)$ be a $n \times n$ square matrix. We suppose that $\tilde{A} = [\tilde{a}_{i,j}]$ denotes the A transformed by the elementary transformations independently of the times of the operations.

Algorithm 2 (Fraction-free similar transformation)

$p \leftarrow 0$; % p is the number of the companion blocks.
$k \leftarrow 1$; % k is the index of the pivot element $\tilde{a}_{k+1,k}$.
$qindx \leftarrow 0$; % $qindx$ is the index of the array preserving the denominators.
$rfl \leftarrow 0$; % rfl is the flag whether the fraction-free computation is
% carried out or not.
% These initial values are set up once at the beginning of the computation.

$p \leftarrow p + 1$;
$kk \leftarrow k$; % kk is the index of the principal element of C_p in $\tilde{A}$.
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 { $qindx \leftarrow qindx + 1$;
$\alpha \leftarrow \tilde{a}_{k+1,k}$;
$q(qindx) \leftarrow \alpha$;
if $k + 1 \neq n$ then $\tilde{A} \leftarrow \tilde{A} * \alpha$;
$op2(k + 1, \alpha)$ } ;
else $op2(k + 1, \tilde{a}_{k+1,k})$;
for $i \leftarrow 1$ to k do $op3(i, k + 1, -\tilde{a}_{i,k}/\tilde{a}_{k+1,k})$;
for $i \leftarrow k + 2$ to n do $op3(i, k + 1, -\tilde{a}_{i,k}/\tilde{a}_{k+1,k})$;
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 is the size of the pth companion block.
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 $qindx \leftarrow 0$;

Applying the above algorithm successively, we construct the companion blocks one by one. If one of the companion blocks does not satisfy the condition $\phi_{C_{j+1}}(\lambda) \mid \phi_{C_j}(\lambda)$ in theorem 2.1, C_j is not a true companion block and we have to compute the block C_j again after some appropriate elementary transformations(Kan and Iri, 1982). In such a case, the rule of cancellation of common denominators does not hold any longer. Therefore, we set the flag $rfl \leftarrow 1$ and let **Algorithm 2** use the straightforward similar transformation after that point. Just before the second computation, the intermediate matrix $\tilde{A}$ has the following form. Some of $\star$s are non-zero and they are not always polynomials.

$$\tilde{A} = \left[\begin{array}{cccc|ccc|cccc}
0 & \cdots & 0 & c_{1,1} & & & & & & & \star \\
1 & \ddots & \vdots & c_{1,2} & & & & & & & \vdots \\
& \ddots & 0 & \vdots & & O & & & & & \vdots \\
O & & 1 & c_{1,m_1} & & & & & & & \vdots \\
\hline
& & & & & \ddots & & & & & \star \\
\hline
& & & & & & & 0 & \cdots & 0 & c_{p,1} \\
& O & & & & & & 1 & \ddots & \vdots & c_{p,2} \\
& & & & & & & & \ddots & 0 & \vdots \\
& & & & & & & O & & 1 & c_{p,m_p}
\end{array}\right] \tag{3.4}$$

When we restart the elimination, we have to put out the common denominators of rational expressions $\star$s in oreder to keep the computation fraction-free. However, such putting out causes the swell of elements, because the denominator does not cancel until the end of the elimination. Therefore, we adopt the straightforward fraction-producing algorithm in such a case.

4. Experimental results and discussion

4.1. EXPERIMENTAL RESULTS

We have implemented the both of above-mentioned algorithms on computer algebra system Reduce 3.5. Using SUN SPARC server 10 (36MHz) with 32MB memory, we computed rational normal forms of matrices of sizes from 3 to 20 of which elements are univariate polynomials with integer coefficients. The matrices have elementary divisors L_N $(N = 3, \ldots, 20)$ as follows.

[Data1] Rational normal forms which consist of one companion block $(R = C_1)$

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

[Data2] Rational normal forms which consist of three companion blocks $(R = C_1 \oplus C_2 \oplus C_3)$

(We suppose that $f_1 = \lambda - 3x,\ f_2 = \lambda - 5x,\ f_3 = \lambda - 7x,\ f_4 = \lambda - 11x$.)

$L_3 = \{f_1, f_1, f_1\},\ L_4 = \{{f_1}^2, f_1, f_1\},\ L_5 = \{{f_1}^2, {f_1}^2, f_1\},\ L_6 = \{{f_1}^2, {f_1}^2, {f_1}^2\},$
$L_7 = \{{f_1}^3, {f_1}^2, {f_1}^2\},\ L_8 = \{{f_1}^3, {f_1}^3, {f_1}^2\},\ L_9 = \{{f_1}^3, {f_1}^3, {f_1}^3\},$
$L_{10} = \{{f_1}^3 f_2, {f_1}^3, {f_1}^3\},\ L_{11} = \{{f_1}^3 f_2, {f_1}^3 f_2, {f_1}^3\},\ L_{12} = \{{f_1}^3 f_2, {f_1}^3 f_2, {f_1}^3 f_2\},$
$L_{13} = \{{f_1}^3 {f_2}^2, {f_1}^3 f_2, {f_1}^3 f_2\},\ L_{14} = \{{f_1}^3 {f_2}^2, {f_1}^3 {f_2}^2, {f_1}^3 f_2\},$
$L_{15} = \{{f_1}^3 {f_2}^2, {f_1}^3 {f_2}^2, {f_1}^3 {f_2}^2\},\ L_{16} = \{{f_1}^3 {f_2}^2 f_3, {f_1}^3 {f_2}^2, {f_1}^3 {f_2}^2\},$
$L_{17} = \{{f_1}^3 {f_2}^2 f_3, {f_1}^3 {f_2}^2 f_3, {f_1}^3 {f_2}^2\},\ L_{18} = \{{f_1}^3 {f_2}^2 f_3, {f_1}^3 {f_2}^2 f_3, {f_1}^3 {f_2}^2 f_3\},$
$L_{19} = \{{f_1}^3 {f_2}^2 f_3 f_4, {f_1}^3 {f_2}^2 f_3, {f_1}^3 {f_2}^2 f_3\},\ L_{20} = \{{f_1}^3 {f_2}^2 f_3 f_4, {f_1}^3 {f_2}^2 f_3 f_4, {f_1}^3 {f_2}^2 f_3\}$

The practical input matrices data were made by the following process.

(a) Construct the rational normal forms C_N by theorem 2.2, which correspond to the elementary divisors assumed to be the results.

Table 1. CPU-Time(msec) for the computation of rational normal forms for Data1 (Each of rational normal forms consists of one companion block.)

size(n)	straight(t_1)	frac-free(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

(b) Prepare regular matrices for randomization.

$$S_n = [s_{i,j}], \;\; s_{i,j} = \begin{cases} n+1-\max(i,j) & i \geq j-1 \\ 0 & i < j-1 \end{cases}$$

(c) Apply the similar transformations to C_N, and obtain $A_N = S_N C_N {S_N}^{-1}$.
Since $\det S_N = 1$, all the elements of each A_N are univariate polynomials with integer coefficients.

We show the CPU-times (in milli-seconds) and the ratios between **Algorithm 1** and **Algorithm 2**. Table 1 shows the results for Data1 and Table 2 for Data2. As we mentioned in section 2, we can compute the transformation matrices simultaneously. When we compute including S and S^{-1}, their CPU-times are almost double of those in the tables.

Table 2. CPU-Time(msec) for the computation of rational normal forms for Data2 (Each of rational normal forms consists of three companion blocks.)

size(n)	straight(t_1)	frac-free(t_2)	t_2/t_1	size of the biggest block C_1
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	

4.2. DISCUSSION

Table 1 and Table 2 show that all the ratios of the CPU-times are less than 1 except the case for $n = 3, 4$. For small-sized matrices, the overhead to avoid rational expressions is considered to overcome the effect of fraction-free computation. However, for larger-sized matrices, the fraction-free algorithm turns out to be more efficient than the straightforward one.

In Table 1, the ratio gradually decreases down to 0.27 in inverse proportion to the size of matrices. For the sizes 19 and 20, the ratio slightly shifts to the increase. The result of another experiment shows similar tendency and the ratio stays around 0.28 up to $n = 25$. Although the efficiency depends on the elements of input matrices and their arrangement, we consider that the effect of fraction-free computation is confirmed.

Table 2 shows that the size of the biggest companion block C_1 is an important factor in the efficiency. When the size of C_1 increases at $n = 4, 7, 10, 13, 16, 19$, the ratio increases. In each group of the same size of C_1, the ratio gradually decreases in inverse proportion to n.

All the examples in above data keep fraction-free computation through, because none of them causes the recomputation of true companion blocks. Other experiments for such types of matrices are needed in the future, but it is difficult in itself to make such examples artificially.

5. Conclusion

In this papaer, we developed a fraction-free algorithm for computing rational normal forms, in order to avoid the explosive swell of rational expressions, by guessing from Bareiss' fraction-free elimination. We have implemented it on the computer algebra system Reduce3.5, and computed rational normal forms of some examples. The experimental results show that our algorithm is more efficient than the traditional one.

Although we mentioned the only case where the elements of matrices are univariate polynomials, the cancellation rule holds also for integer matrices. For integer matrices, some efficient algorithms using modular arithmetic have been developed (Giesbrecht, 1995), but it seems difficult to extend such methods directly to polynomial matrices.

Single-step fraction-free Gaussian elimination can be modified by reducing its redundancy and putting several steps together. Then we obtain multistep integer-preserving Gaussian elimination(Bareiss, 1968). However, for the case of our algorithm in similar transformation, there seems no such possibility of improvement, because we could not find the determinant expression of elimination process like Sylvester's Identity.

Computer algebra systems Reduce3.5 has already a matrix normal form package contributed by users. The programs in this package could not compute all the normal forms for the same data as this paper. They required much more CPU-time and memory than our programs. Therefore, we consider that our programs are sufficiently practical.

Acknowledgements

We are grateful to Prof. Hoon Hong for pointing out the related works and for the valuable discussions.

References

Bareiss, E. H. (1968). Sylvester's Identity and Multistep Integer Preserving Gaussian Elimination. *Math.Comp.*, **22**(103):565–578.

Collins, G. E. (1966). Polynomial Remainder Sequences and Determinants. *Amer.Math.Monthly*, **73**(7):708–712.

Collins, G. E. (1967). Subresultants and Reduced Polynomial Sequences. *J.ACM*, **14**(1):128–142.

Danilevskii, A. M. (1937). On the numerical solution of the secular equation. *Mat.Sb.*, **2**(1):169–172. (in Russian).

Gantmacher, F. R. (1959). *The Theory of Matrices (2nd ed.)*, volume 1. Chelsea.

Giesbrecht, M. (1995). Nearly Optimal Algorithms for Canonical Matrix Forms. *SIAM J.Comput.*, **24**(5):948–969.

Gil, I. (1992). Computation of the Jordan Canonical Form of a Square Matrix (Using the Axiom Programming Language). In *ISSAC '92*, pages 138–145.

Kaltofen, E., Krishnamoorthy, M. S., Saunders, B. D. (1990). Parallel Algorithms for Matrix Normal Forms. *Linear Algebra Its Appl.*, **136**:189–208.

Kan, T., Iri, M. (1982). *Jordan Normal Form.* Univ. of Tokyo Press. (in Japanese).

Kuriyama, K., Moritsugu, S. (1994). Exact Computation of Normal Forms of Matrices by Computer Algebra. *Trans.Japan Soc.Indust.Appl.Math.*, **4**(2):183–194. (in Japanese).

Kuriyama, K., Moritsugu, S. (1996). Fraction-free Method for Computing Rational Normal Forms of Square Matrices. *Trans.Japan Soc.Indust.Appl.Math.*, **6**(3):253–264. (in Japanese).

Moritsugu, S., Kuriyama, K. (1992). Exact Computation of Jordan Normal Forms of Matrices by Computer Algebra. *Trans.Japan Soc.Indust.Appl.Math.*, **2**(1):91–103. (in Japanese).

Rouillier, F. (1995). Bareiss Formulas and Reduction of Quadratic Forms. *Compt.Rend.Acad.Sci.Ser.I-Math.*, **320**(10):1273–1277. (in French).

Sasaki, T., Murao, H. (1982). Efficient Gaussian Elimination Method for Symbolic Determinants and Linear Systems. *ACM Trans.Math.Softw.*, **8**(3):277–289.

Takeshima, T., Yokoyama, K., Noro, M. (1991). A Method for Computing Rational Normal Forms of Matrices. In *Applications of Computer Algebra to Mathematical Research*, volume 753 of *Papers Res.Inst.Math.Sci., Kyoto Univ.*, pages 277–289. (in Japanese).

A. Appendix

In place of the proof of (3.2), we show the process of straightforward algorithm, representing the sixteen elements of 4×4 square matrix by sixteen distinct variables. Though too long expressions are abbreviated in the following results, the formulae have been confirmed by symbolic computation. In the general case where the size is more than five, the elements under the 4th row and righter than the 4th column behave in the same way as the elements of the 4th row and column. Therefore, it is enough to show the case of size four. We suppose that $\bar{A}^{(k)} = \left[\bar{a}_{i,j}^{(k)}\right]$ denotes the transformed matrix A whose kth column is eliminated.

We start from the matrix

$$A = \begin{bmatrix} a_{1,1} & a_{1,1} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \end{bmatrix}. \tag{A.1}$$

Applying the operations $(op2(2, a_{2,1}), op3(1, 2, -a_{1,1}), op3(3, 2, -a_{3,1}),$ $op3(4, 2, -a_{4,1}))$ to eliminate the 1st column using $a_{2,1}$ as a pivot, we get

$$\bar{A}^{(1)} = \begin{bmatrix} 0 & \frac{\triangle}{a_{2,1}} & \frac{\triangle}{a_{2,1}} & \frac{\triangle}{a_{2,1}} \\ 1 & \frac{\triangle}{a_{2,1}} & \frac{\triangle}{a_{2,1}} & \frac{\triangle}{a_{2,1}} \\ 0 & \frac{\beta_{3,2}}{a_{2,1}} & \frac{\triangle}{a_{2,1}} & \frac{\triangle}{a_{2,1}} \\ 0 & \frac{\triangle}{a_{2,1}} & \frac{\triangle}{a_{2,1}} & \frac{\triangle}{a_{2,1}} \end{bmatrix}, \tag{A.2}$$

where $\triangle$s and $\beta_{3,2}$ are the polynomials not divisible by $a_{2,1}$. At this moment after the elimination of the 1st column, the elements righter than the 1st column include $a_{2,1}$ as their common denominator.

Applying the operations $(op2(3, \beta_{3,2}/a_{2,1})$, $op3(1, 3, -\bar{a}_{1,2}^{(1)})$, $op3(2, 3, -\bar{a}_{2,2}^{(1)})$, $op3(4, 3, -\bar{a}_{4,2}^{(1)}))$ to eliminate the 2nd column using $\beta_{3,2}/a_{2,1}$ as a pivot, we get

$$\bar{A}^{(2)} = \begin{bmatrix} 0 & 0 & \frac{\triangle}{\beta_{3,2}} & \frac{\triangle}{\beta_{3,2}} \\ 1 & 0 & \frac{\triangle}{\beta_{3,2}} & \frac{\triangle}{\beta_{3,2}} \\ 0 & 1 & \frac{\triangle}{\beta_{3,2}} & \frac{\triangle}{\beta_{3,2}} \\ 0 & 0 & \frac{\triangle}{\beta_{3,2}} & \frac{\triangle}{\beta_{3,2}} \end{bmatrix}. \tag{A.3}$$

At this moment after the elimination of the 2nd column, the elements righter than the 2nd column include $\beta_{3,2}$ as their common denominator but $a_{2,1}$ does not appear in the denominator.

If we apply $op2(3, a_{2,1})$ to the above matrix $\bar{A}^{(2)}$, then we get $A^{(2)}$ (3.2) in the text; hence we have shown that $A^{(2)}$ (3.2) is correct. □