Regular Paper

Evaluation of Symbolic Expression of the First Derivative of Determinant

HIROKAZU MURAO*, HIDETSUNE KOBAYASHI** and TETSURO FUJISE***

A method to evaluate the first derivative of the determinant of a given matrix of symbolic elements is considered. If evaluations such as substitution of numbers or a parameter for variables reduce the total number of variables, the determinant expansion can be made efficient by performing such evaluation elementwise before expansion, however, if the derivative of the determinant is required, we cannot perform such substitution for the differentiation variable in each matrix element. In this paper, we introduce a new operator ***ddet*** in order to solve this problem, and express these computational processes in terms of the operator. Given two matrices, the ***ddet*** operator computes the sum of the determinants of matrices constructed from the elements of both matrices, and, specially if one of the matrices is of derivative elements of the other, gives the first derivative of the determinant of the matrix. Also presented in this paper are efficient algorithms for ***ddet*** in terms of inductive formulas, where the recalculations of common subexpressions are eliminated. The algorithms are natural extensions of the known algorithms; minor expansion, Bareiss's Gaussian elimination, and, for polynomial cases, the interpolation algorithms. Finally, as an empirical study, we give a model of matrices, which appear when solving a system of algebraic equations by a general elimination method, and indicate the choice of the algorithm suited for our particular application, showing the effectiveness of our new operator as well.

1. Introduction

Expansion of symbolic determinants requires far more extra memory space than the input matrices and their results, as well as a large amount of computing time, in general, due to growths of symbolic expressions. For the determinant expansion, both Gaussian elimination and minor expansion algorithms are used. The former implies the elementwise expression growth due to its computational nature with division, especially in multivariate cases, and the latter must treat common subexpressions of minors, which leads to a significant problem as large matrices are treated. To overcome these difficulties and to improve efficiency, various techniques were invented [2-4, 11, 13]. Besides the growth of intermediate expressions, there can be considerable growth of the final results. This is distinct when elements are multivariate, and the number of variables heavily affects on the expression growth. Therefore, when some value of a determinant is required, *e.g.* evaluation of variables with some numerical values or parameterization of every variable with a single parameter, which are expected to reduce the number of variables it is desirable to perform such substitution elementwise and then to expand the determinant with replaced elements.

We now consider the evaluation of the first derivative of a symbolic determinant. Let M and M_{ij} denote an $N \times N$ square matrix and its i-j element respectively. Suppose we compute a parameterization by t of $\partial/\partial U_k \det(M_{ij}(U_0, \ldots, U_n))$ obtained by substituting $(1, a_1 t, \ldots, a_n t)$ for $(U_0, U_1, \ldots, U_n)$, where a_i's are some numerical values and U_i's are independent variables (thus, $\partial/\partial U_k$ can be replaced by $\mathrm{d}/\mathrm{d}U_k$).[1] In this case, we cannot perform the above mentioned computational method, *i.e.* substitution before determinant expansion, for the variable of differentiation U_k. However, as was noted in [7], the derivative can be expanded, using the distribution law of differentiation, as

$$\frac{\mathrm{d}}{\mathrm{d}U_k}\det(M_{ij}) = \sum_{j=1}^{N} \det \begin{pmatrix} M_{11} & \cdots & \dfrac{\mathrm{d}M_{1j}}{\mathrm{d}U_k} & \cdots & M_{1N} \\ \vdots & & \vdots & & \vdots \\ M_{N1} & \cdots & \dfrac{\mathrm{d}M_{Nj}}{\mathrm{d}U_k} & \cdots & M_{NN} \end{pmatrix}, \tag{1}$$

and then, the substitution of the parameter can be performed also for U_k in each element of each matrix in the sum before determinant expansion. Using this method, we can reduce the original problem of $n+1$ variables to

*Computer Centre, University of Tokyo, Yayoi 2-11-16, Bunkyo-ku, Tokyo 113, Japan.
**Department of Mathematics, College of Science and Technology, Nihon University, Kanda-Surugadai 1-8, Chiyoda-ku, Tokyo 101, Japan.
***Mitsubishi Research Institute, Inc., Otemachi 2-3, Chiyoda-ku, Tokyo 100, Japan. Currently, ICOT, Mita Kokusai Bldg. 21F, 4-28, Mita 1-chome, Minato-ku, Tokyo 108, Japan.

[1] This kind of calculations actually appear when factorizing U-resultant [7].

much simpler problem of univariate determinants, which could improve the computing time and suppress the expression growth very much, enabling us to treat larger problems of matrices with larger dimensions.[1]

Another example shows the benefit of our computing method more clearly. We compute the value of

$$\frac{\mathrm{d}}{\mathrm{d}z}\begin{vmatrix}(1+z)e^{z^2} & z+z\sin(z)\\ z^2e^z & z+\cos^2(z)\end{vmatrix}$$

at $z=0$. Because in computer algebra system, (transcendental) functions are treated as independent variables in arithmetic operations, the calculation of the determinant is equivalent to those of multivariate cases (5 variables in this example), although the evaluation at $z=0$ reduces the elements and their derivatives to integers. Then, again by using (1), the value required can be computed efficiently, as a sum of two determinants of numerical entries:

$$\begin{aligned}&\frac{\mathrm{d}}{\mathrm{d}z}\det\begin{pmatrix}(1+z)e^{z^2} & z+z\sin(z)\\ z^2e^z & z+\cos^2(z)\end{pmatrix}\Bigg|_{z=0}\\ &=\det\begin{pmatrix}e^{z^2}+(1+z)\cdot 2ze^{z^2} & z+\sin(z)\\ 2ze^z+z^2e^z & z+\cos^2(z)\end{pmatrix}\Bigg|_{z=0}\\ &\quad+\det\begin{pmatrix}(1+z)e^{z^2} & 1+\sin(z)+z\cos(z)\\ z^2e^z & 1-2\cos(z)\sin(z)\end{pmatrix}\Bigg|_{z=0}\\ &=\det\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}+\det\begin{pmatrix}1 & 1\\ 0 & 1\end{pmatrix}=2.\end{aligned}$$

Now the merit of our computational method by (1) is very clear for symbolic matrices, and one should immediately notice that expansions of the determinants in the sum yield many common subexpressions independently of which algorithm to use for determinant expansion. In order to facilitate this method and to avoid recalculation of those common subexpressions, we introduce a new operator ***ddet***.

Definition of *ddet*: Given two $N\times N$ matrices M and Δ, ***ddet*** (M, Δ) is defined as follows:

$$\textit{ddet}\,(M,\Delta)\equiv\sum_{s=1}^{N}\det({}^{(s)}D) \tag{2}$$

where ${}^{(s)}D(1\le s\le N)$ is also an $N\times N$ matrix defined by

$${}^{(s)}D_{ij}=\begin{cases}\Delta_{is} & \text{if } j=s\\ M_{ij} & \text{otherwise}\end{cases}\qquad 1\le i\le N. \tag{3}$$

We do not consider algebraic properties of ***ddet*** but only note that

$$\textit{ddet}\,(M,\Delta)=\frac{\mathrm{d}}{\mathrm{d}z}\det(M)\ \text{if}\ \Delta_{ij}=\frac{\mathrm{d}M_{ij}}{\mathrm{d}z}.$$

The algorithms to realize the ***ddet*** operator can be given as natural extensions of the known algorithms for symbolic determinant, minor expansion and Gaussian elimination, as described in the later sections, based on the distribution formula (1). In those algorithms, the determinants in the sum are treated all together and their common subexpressions are constructed from the elements of M and Δ. In other words, the value of ***ddet*** (M, Δ) is defined and computed as a combination of the elements of M and Δ. Thus, if the arguments are matrices of evaluated elements, then the ***ddet*** operator computes the evaluated expression of ***ddet*** (M, Δ). This enables us to compute the evaluated expression of the first derivative of a determinant very efficiently.

Now that we devised an algorithm of ***ddet*** for numerical cases, we can apply the interpolation algorithm to ***ddet*** calculation for the polynomial cases; values for interpolation can be computed by the ***ddet*** operator with two matrix arguments in which some numerical values are substituted for the differentiation variable z.

In the sections to follow, we shall present the concrete descriptions of algorithms by means of mathematical formulas of induction for clarity and compactness. Treated are the nested minor expansion algorithm (abbreviated by NME) [2, 3, 12], Bareiss's integer-preserving Gaussian elimination algorithm (one-step and two-step, abbreviated by GE1 and GE2 respectively), and, for polynomial cases, the interpolation algorithm.

2. Nested Minor Expansion Algorithm for *ddet*

Direct expansion of determinants by minors generates a lot of common subexpressions of minors as the expansion steps proceed to smaller minors. The easiest way to treat this problem of common subexpressions is to prepare a table to store the subexpressions [3, 4], but this requires an exponential amount of extra space for the table at worst. Therefore we do not consider such an algorithm here. Another way to eliminate recalculation of those subexpressions is to construct all the minors step by step from the smallest, *i.e.* 2×2 minors, systematically. This algorithm is called nested minor expansion, and is well described by the following iterative formulas:

$$\begin{aligned}&\sigma^{(1)}_{\{s_1\}}=M_{1s_1}\quad\text{for}\quad 1\le s_1\le N,\\ &\sigma^{(i+1)}_{\{s_1\cdots s_{i+1}\}}=\sum_{j=1}^{i+1}(-1)^{i+j+1}\cdot M_{i+1s_j}\cdot\sigma^{(i)}_{\{s_1\cdots s_{i+1}\}\setminus s_j},\end{aligned} \tag{4}$$

where $\{s_1 \ldots s_k\}$ represents a set of k integers s.t. $1\le s_1<s_2<\cdots<s_k\le N$, and $\{s_1 \ldots s_k\}\setminus s_j$ denotes the set $\{s_1 \ldots s_k\}$ without an integer $s_j\in\{s_1 \ldots s_k\}$. The expressions $\sigma^{(i)}$ correspond to a set of $i\times i$ minors, and $\det(M)$ is given by $\det(M)=\sigma^{(N)}_{\{1\ldots N\}}$.

Now let us consider the nested minor expansion algorithm for ***ddet*** given by (2). The determinants in the sum can be expanded in parallel, however the expansion yields a lot of common subexpressions of minors to ${}^{(s)}D$'s; every $\sigma^{(i)}_{\{s_1\ldots s_i\}}$ is common for all ${}^{(s)}D$'s s.t. $s\notin$

[1] Of course, U_i's $(i\neq k)$ can be replaced by the expressions containing a parameter irrelevant to differentiation in this example. Even after this pre-substitution, the number of variables in the matrix can be reduced by 1 with our method, which is beneficial to determinant calculation.

$\{s_1 \ldots s_k\}$. By expanding the determinants in the sum all at once, we can eliminate recalculation of those common subexpressions. On the way to calculate the minors of ${}^{(s)}D$'s, we classify them into two classes by whether to contain an element of Δ or not. The one is a sequence of $\sigma^{(i)}$ defined above, which does not contain any element of Δ. Another sequence, denoted by $\tau^{(i)}$, is defined to be a set of $i \times i$ minors with a single column from Δ included

$$\tau^{(1)}_{\{s_1\}} = \Delta_{1s_1} \quad \text{for} \quad 1 \le s_1 \le N,$$

$$\tau^{(i+1)}_{\{s_1 \ldots s_{i+1}\}} = \sum_{j=1}^{i+1} (-1)^{i+j+1} \cdot [M_{i+1s_j} \cdot \tau^{(i)}_{\{s_1 \ldots s_{i+1}\} \setminus s_j} + \Delta_{i+1s_j} \cdot \tau^{(i)}_{\{s_1 \ldots s_{i+1}\} \setminus s_j}] \quad 1 \le i \le N-1.$$

Then, the sequence $\tau^{(i)}$ results in giving

$$\textit{ddet}\,(M, \Delta) = \tau^{(N)}_{\{1 \ldots N\}}.$$

3. Gaussian Elimination with Exact Division

Gaussian elimination can be performed without normalizing pivots but with exact divisions by Bareiss's algorithm. In 1968, Bareiss showed [1] that pivot at some elimination step exactly divides every element after the next elimination step. This exact division can be postponed by any number of elimination steps, and this defines the multi-step algorithms. In this paper, we treat only the one-step and the-two-step cases. Throughout this section, $M^{(i+1)}$ denotes the matrix of M obtained after the i-th elimination step in general; $M^{(1)}$ corresponds to the matrix M itself with no elimination performed.

Consider the calculations of det (${}^{(s)}D$)'s by Gaussian elimination. From the definition (3), the subexpressions ${}^{(s)}D^{(i)}_{kj}$ obtained during elimination are common for all $s \le i$ and $i \le k$, $j \le N$ except for $j=s$, but for $s>i$, ${}^{(s)}D^{(i)}_{kj}$ differs from each other because it includes elements from Δ. Therefore we perform elimination on each ${}^{(s)}D$ in two phases. The first phase treats the common subexpressions derived from M_{kj} in a single sequence of elimination steps for all ${}^{(s)}D$. During the first phase, expressions derived from Δ are also calculated using the same elimination process for the elements of M. For each $s(1 \le s \le N)$, once the elimination step i reaches s, the expressions which constitute ${}^{(s)}D^{(s)}$ are collected to form a square matrix of dimension $N-s+1$, and the matrix is dispatched to the normal Gaussian elimination for obtaining the determinant det ${}^{(s)}D$, which is the second phase of our algorithms. We formulate this two-phase process for both Bareiss's one-step and two-step algorithms. We define the initial values as $M^{(0)}_{00}=1$, as well as $M^{(1)} \equiv M$ and $\Delta^{(1)} \equiv \Delta$. Figure 1 shows the flow of expressions in each phase of our algorithms. Here we only note that the calculations of the second phase can be performed completely in parellel, as can be seen in the figure.

3.1 Bareiss's One-step Algorithm for *ddet*

The one-step algorithm proceeds eliminating column by column. In the first phase, M and Δ are transformed with each column of M being eliminated. This process is described by the following iteration formulas:

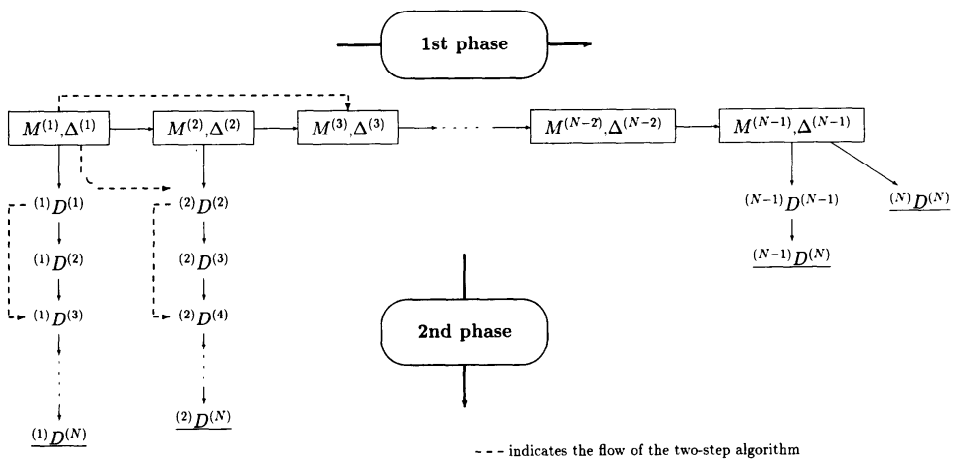

Fig. 1 Flow of expressions in Bareiss's Gaussian elimination algorithm for ***ddet***.

$$M_{kj}^{(i+1)}=\begin{vmatrix} M_{ii}^{(i)} & M_{ij}^{(i)} \\ M_{ki}^{(i)} & M_{kj}^{(i)} \end{vmatrix} \Big/ M_{i-1i-1}^{(i-1)}=\frac{M_{ii}^{(i)}M_{kj}^{(i)}-M_{ki}^{(i)}M_{ij}^{(i)}}{M_{i-1i-1}^{(i-1)}} \quad \text{for} \quad 2\le i+1\le k,\ j<N,$$

$$\Delta_{kj}^{(i+1)}=\begin{vmatrix} M_{ii}^{(i)} & \Delta_{ij}^{(i)} \\ M_{ki}^{(i)} & \Delta_{kj}^{(i)} \end{vmatrix} \Big/ M_{i-1i-1}^{(i-1)}=\frac{M_{ii}^{(i)}\Delta_{kj}^{(i)}-M_{ki}^{(i)}\Delta_{ij}^{(i)}}{M_{i-1i-1}^{(i-1)}} \quad \text{for} \quad 2\le i+1\le k,\ j\le N. \tag{5}$$

At each step $s(1\le s\le N-1)$, we construct an $(N-s+1)\times(N-s+1)$ matrix ${}^{(s)}D^{(s)}$ by collecting the first column from $\Delta^{(s)}$ and the rest from $M^{(s)}$:

$${}^{(s)}D_{kj}^{(s)}=\begin{cases} \Delta_{k+s-1s}^{(s)} & \text{for } j=1, \\ M_{k+s-1j+s-1}^{(s)} & \text{otherwise,} \end{cases} \quad \text{for} \quad 1\le k,\ j\le N-s+1.$$

Letting the initial divisor (pivot) of the elimination be the last pivot of M, *i.e.*, ${}^{(s)}D_{00}^{(s-1)}=M_{s-1s-1}^{(s-1)}$, we start the second phase to apply the one-step algorithm for determinant to ${}^{(s)}D^{(s)}$:

$${}^{(s)}D_{kj}^{(s+i)}=\begin{vmatrix} {}^{(s)}D_{ii}^{(s+i-1)} & {}^{(s)}D_{ij}^{(s+i-1)} \\ {}^{(s)}D_{ki}^{(s+i-1)} & {}^{(s)}D_{kj}^{(s+i-1)} \end{vmatrix} \Big/ {}^{(s)}D_{i-1i-1}^{(s+i-2)}$$
$$=\frac{{}^{(s)}D_{ii}^{(s+i-1)}\cdot{}^{(s)}D_{kj}^{(s+i-1)}-{}^{(s)}D_{ki}^{(s+i-1)}\cdot{}^{(s)}D_{ij}^{(s+i-1)}}{{}^{(s)}D_{i-1i-1}^{(s+i-2)}}$$
$$\text{for} \quad 2\le i+1\le k,\ j\le N-s+1,$$

to obtain $\det({}^{(s)}D)={}^{(s)}D_{NN}^{(N)}$ $(1\le s\le N-1)$. Besides this sequence of the second phase, $\det({}^{(N)}D)$ is given by $\Delta_{NN}^{(N)}$ as the final expression of (5). And finally we obtain all the determinants $\det({}^{(s)}D)$ $(1\le s\le N)$ to give

$$\textit{ddet}\,(M,\Delta)=\sum_{s=1}^{N}{}^{(s)}D_{NN}^{(N)}.$$

3.2 Bareiss's Two-step Algorithm for *ddet*

We extend the two-step algorithm to the one for ***ddet*** (M, Δ) computation. The two-step algorithm is to be expressed in parallel with the one-step algorithm, except that two matrices ${}^{(s)}D^{(s)}$ and ${}^{(s+1)}D^{(s)}$ are dispatched to the second phase at each step s. The two-step version of Bareiss's algorithm is defined simply by

$$M_{kj}^{(i+1)}=\begin{vmatrix} M_{ii}^{(i)} & M_{ii+1}^{(i)} & M_{ij}^{(i)} \\ M_{i+1i}^{(i)} & M_{i+1i+1}^{(i)} & M_{i+1j}^{(i)} \\ M_{ki}^{(i)} & M_{ki+1}^{(i)} & M_{kj}^{(i)} \end{vmatrix} \Big/ M_{i-1i-1}^{(i-1)2}, \quad 3\le i+2\le k,\ j<N,$$

but the actual algorithm works as follows:

$$M_{kj}^{(i+1)}=\frac{M_{i+1i+1}^{(i+1)}M_{kj}^{(i)}-M_{ki+1}^{(i+1)}M_{i+1j}^{(i)}+M_{ki+1}^{(i+1)\prime}M_{ij}^{(i)}}{M_{i-1i-1}^{(i-1)}} \tag{6}$$

where $M_{i+1i+1}^{(i+1)}$, $M_{ki+1}^{(i+1)}$ and $M_{ki+1}^{(i+1)\prime}$ are the subexpressions common to multiple k-j pairs and defined by

$$M_{i+1i+1}^{(i+1)}=\begin{vmatrix} M_{ii}^{(i)} & M_{ii+1}^{(i)} \\ M_{i+1i}^{(i)} & M_{i+1i+1}^{(i)} \end{vmatrix} \Big/ M_{i-1i-1}^{(i-1)},$$

$$M_{ki+1}^{(i+1)}=\begin{vmatrix} M_{ii}^{(i)} & M_{ii+1}^{(i)} \\ M_{ki}^{(i)} & M_{ki+1}^{(i)} \end{vmatrix} \Big/ M_{i-1i-1}^{(i-1)},$$

and

$$M_{ki+1}^{(i+1)}=\begin{vmatrix} M_{i+1i}^{(i)} & M_{i+1i+1}^{(i)} \\ M_{ki}^{(i)} & M_{ki+1}^{(i)} \end{vmatrix} \Big/ M_{i-1i-1}^{(i-1)}.$$

And for Δ,

$$\Delta_{kj}^{(i+1)}=\frac{M_{i+1i+1}^{(i+1)}\Delta_{kj}^{(i)}-M_{ki+1}^{(i+1)}\Delta_{i+1j}^{(i)}+M_{ki+1}^{(i+1)\prime}\Delta_{ij}^{(i)}}{M_{i-1i-1}^{(i-1)}} \quad \text{for} \quad 3\le i+2\le k,\ j\le N.$$

At each step $s(1\le s<N-1)$, two $(N-s+1)\times(N-s+1)$ matrices ${}^{(s)}D^{(s)}$ and ${}^{(s+1)}D^{(s)}$ are constructed from the elements of $M^{(s)}$ and $\Delta^{(s)}$ as

$$ {}^{(s)}D_{kj}^{(s)} = \begin{cases} \Delta_{k+s-1s}^{(s)} & \text{for } j=1, \\ M_{k+s-1j+s-1}^{(s)} & \text{otherwise,} \end{cases} \quad \text{for} \quad 1 \le k, j \le N-s+1, $$

$$ {}^{(s)}D_{kj}^{(s+1)} = \begin{cases} \Delta_{k+s-1s}^{(s)} & \text{for } j=2, \\ M_{k+s-1j+s-1}^{(s)} & \text{otherwise,} \end{cases} \quad \text{for} \quad 1 \le k, j \le N-s+1, $$

and are dispatched to the two-step elimination of the second phase with the initial values ${}^{(s)}D_{00}^{(s-1)} = {}^{(s+1)}D_{00}^{(s-1)} = M_{s-1s-1}^{(s-1)}$.

The iteration formula for the second phase is given by substituting ${}^{(s)}D$ for M in (6):

$$ {}^{(t)}D_{kj}^{(s+i+1)} = \frac{{}^{(t)}D_{i+1i+1}^{(s+i)} {}^{(t)}D_{kj}^{(s+i-1)} - {}^{(t)}D_{ki+1}^{(s+i)} {}^{(t)}D_{i+1j}^{(s+i-1)} + {}^{(t)}D_{ki+1}^{(s+i)\prime} {}^{(t)}D_{ij}^{(s+i-1)}}{{}^{(t)}D_{i-1i-1}^{(s+i-2)}} $$

$$ \text{for} \quad t=s, s+1, \text{ and } 3 \le i+2 \le k, j \le N-s+1, $$

where the subexpressions are defined by:

$$ {}^{(t)}D_{i+1i+1}^{(s+i)} = \begin{vmatrix} {}^{(t)}D_{ii}^{(s+i-1)} & {}^{(t)}D_{ii+1}^{(s+i-1)} \\ {}^{(t)}D_{i+1i}^{(s+i-1)} & {}^{(t)}D_{i+1i+1}^{(s+i-1)} \end{vmatrix} \Big/ {}^{(t)}D_{i-1i-1}^{(s+i-2)}, $$

$$ {}^{(t)}D_{ki+1}^{(s+i)} = \begin{vmatrix} {}^{(t)}D_{ii}^{(s+i-1)} & {}^{(t)}D_{ii+1}^{(s+i-1)} \\ {}^{(t)}D_{ki}^{(s+i-1)} & {}^{(t)}D_{ki+1}^{(s+i-1)} \end{vmatrix} \Big/ {}^{(t)}D_{i-1i-1}^{(s+i-2)}, $$

$$ {}^{(t)}D_{ki+1}^{(s+i)\prime} = \begin{vmatrix} {}^{(t)}D_{i+1i}^{(s+i-1)} & {}^{(t)}D_{i+1i+1}^{(s+i-1)} \\ {}^{(t)}D_{ki}^{(s+i-1)} & {}^{(t)}D_{ki+1}^{(s+i-1)} \end{vmatrix} \Big/ {}^{(t)}D_{i-1i-1}^{(s+i-2)}. $$

If N is even, then the steps of the one-step elimination are required to obtain $\det({}^{(s)}D)$:

$$ \det({}^{(N)}D) = \Delta_{NN}^{(N)} = \begin{vmatrix} M_{N-1N-1}^{(N-1)} & \Delta_{N-1N}^{(N-1)} \\ M_{NN-1}^{(N-1)} & \Delta_{NN}^{(N-1)} \end{vmatrix} \Big/ M_{N-2N-2}^{(N-2)}, $$

$$ \det({}^{(N-1)}D) = \begin{vmatrix} \Delta_{N-1N-1}^{(N-1)} & M_{N-1N}^{(N-1)} \\ \Delta_{NN-1}^{(N-1)} & M_{NN}^{(N-1)} \end{vmatrix} \Big/ M_{N-2N-2}^{(N-2)}, $$

$$ \det({}^{(s)}D) = {}^{(s)}D_{NN}^{(N)} = \begin{vmatrix} {}^{(s)}D_{N-1N-1}^{(N-1)} & {}^{(s)}D_{N-1N}^{(N-1)} \\ {}^{(s)}D_{NN-1}^{(N-1)} & {}^{(s)}D_{NN}^{(N-1)} \end{vmatrix} \Big/ {}^{(s)}D_{N-2N-2}^{(N-2)} \quad \text{for} \quad 1 \le s \le N-2. $$

Note that all the divisions are exact in the above formulas. Summing up all the determinants of ${}^{(s)}D$, we obtain

$$ \textit{ddet}(M, \Delta) = \sum_{s=1}^{N} \det({}^{(s)}D). $$

4. Interpolation Algorithm for *ddet*

In determinant calculation, the interpolation algorithm, among other algorithms treated in this paper, works most efficiently, if the determinant is known to be a dense polynomial of reasonably small degree and number of variables [6]. For those cases, the interpolation algorithm is expected to work quite well also in the ***ddet*** calculation. In this section, we consider the interpolation algorithm for ***ddet*** for polynomial cases.

Basically, the structure of the interpolation algorithm is independent of what to compute as a result, except that values for interpolation are to be calculated in accordance with the desired polynomial [8]. For our problem in this paper, we can hardly apply the interpolation algorithm to the calculation of $\mathrm{d}/\mathrm{d}z \det(M)$ unless we can formulate the evaluation of the derivative. The ***ddet*** operator computes such values from two evaluated matrices of the original elements and their derivatives. The actual algorithms for ***ddet*** are presented in the former sections. Having prepared the way to compute ***ddet*** of two numerical matrices (Gaussian elimination algorithm is best suited), we describe the interpolation algorithm for ***ddet*** of polynomial cases.

Assume that $\textit{ddet}(M, \Delta)$ is known to be a polynomial in z of degree bounded by an integer d. Let $\tilde{M}_{[z=k]}$ and $\tilde{\Delta}_{[z=k]}$ respectively denote the matrices of elements M_{ij} and Δ_{ij} in which an integer k is substituted for z. The polynomial $\textit{ddet}(M, \Delta)$ is to be interpolated from its $d+1$ values at $z=0, \cdots, d$, *i.e.* $\textit{ddet}(\tilde{M}_{[z=k]}, \tilde{\Delta}_{[z=k]})$'s. Letting the initial polynomials be as $P^{[0]}(z) = \textit{ddet}(\tilde{M}_{[z=0]}, \tilde{\Delta}_{[z=0]})$ and $L^{[0]}(z) = z$, interpolation proceeds as follows:

$$P^{[k]}(z)=P^{[k-1]}(z)+\frac{\boldsymbol{ddet}\,(\tilde{M}_{[z=z_k]},\,\tilde{\Delta}_{[z=z_k]})-P^{[k-1]}(z_k)}{\prod_{i=0}^{k-1}(z_k-z_i)}\cdot L^{[k-1]}(z)$$

$$=P^{[k-1]}(z)+(\boldsymbol{ddet}\,(\tilde{M}_{[z=z_k]},\,\tilde{\Delta}_{[z=zk]})-P^{[k-1]}(k))\cdot L^{[k-1]}(z)/k!,$$

$$L^{[k]}(z)=(z-z_k)\cdot L^{[k-1]}(z)=(z-k)\cdot L^{[k-1]}(z).$$

After d times iteration, the ***ddet*** polynomial is obtained by

$$P^{[d]}(z)=\boldsymbol{ddet}\,(M,\,\Delta).$$

If the ***ddet*** polynomial is multivariate, we have only to apply this interpolation algorithm recursively to compute the polynomial $\boldsymbol{ddet}\,(\tilde{M}_{[z=k]},\,\tilde{\Delta}_{[z=k]})$ with the variable z eliminated, except when the value is numeric in which case the Bareiss's two-step Gaussian elimination algorithm is to be used.

5. Empirical Tests

All the algorithms for ***ddet*** described in this paper are implemented in the symbolic mode of REDUCE [10], and in its algebraic mode, there is prepared functions, using the ***ddet*** operator internally, to compute the first derivative of the determinant of the argument matrix, with and without functionality of substitution for variables. For empirical studies, we use a *linear polynomial entry model*, in which each element of a given $N\times N$ matrix M is a $(n+1)$-variate $(U_0, \ldots, U_n)$ linear polynomial with integer coefficients. This model corresponds to the matrix $\hat{A}$ used in solving a system of algebraic equations by a general elimination method [7, 9], and the solutions of the system can be derived from the parameterizations given in the introduction of the determinant and its partial derivatives by $\partial/\partial U_k$ $(k=0,\cdots,n)$. The parameterized expression of the partial derivative can be computed using the ***ddet*** operator, operated on two matrices of parameterized elements and of parameterized expressions of the partial derivatives of the elements. By the way, if the solution of the system has a multiplicity $e(>1)$, then we require the higher derivatives by $\partial^e/(\partial U_k \partial U_0^{e-1})$ of the determinant, and in this case we have only to perform ***ddet*** calculation w.r.t. $\partial/\partial U_k$ of matrices of bivariate entries of a parameter t and U_0.

We now further simplify the model of calculation, as an empirical test model of our particular application, to matrices of univariate and bivariate (corresponding to the cases of multiplicity) linear polynomial entries, and we shall show which algorithm is best suited to our model. Furthermore we shall discuss about the merit of our ***ddet*** operator compared with the obvious method of expanding multivariate determinant, later.

With this linear polynomial entry model, the multivariate determinant is a polynomial of total degree N at most, and consequently its partial derivative of $N-1$. Thus, its parameterized expression is a uni/bivariate dense polynomial of total degree $N-1$. In such cases of dense polynomials of reasonably small degrees and number of variables, the interpolation algorithm is expected to be most efficient from the analysis of

Table 1 Timing data in millisecond to compute the first derivative of the determinant of matrices of linear polynomial entries with random integer coefficients using ***ddet*** operator. They are measured in REDUCE-2 on Cambridge Lisp running on HITAC M-680H VOS3/ES1 (compatible with IBM 370 MVS/XA, about 35MIPS) with 7MB available memory. NME, GE1, GE2 and INTP stand for the used algorithms; the nested minor expansion, Bareiss's one/two-step Gaussian elimination, and the interpolation algorithm respectively, and **t.o.** indicates time-out ($\geq$600 seconds).

N	univariate				bivariate			
	NME	GE1	GE2	INTP	NME	GE1	GE2	INTP
3	9	7	8	7	12	12	12	20
4	19	28	19	12	36	77	51	55
5	49	98	82	26	114	368	323	160
6	134	281	219	56	376	1364	1075	378
7	356	705	593	114	1161	4327	3814	930
8	928	1601	1280	219	3446	11969	9743	2093
9	2431	3246	2721	415	9643	29480	25369	4448
10	6154	6254	5045	725	26534	67151	55480	8416
11	15181	11222	9249	1221	69674	143738	122600	15837
12	36341	19443	15876	1936	180470	286102	237227	27290
13	87093	32132	26241	3101	t.o.	t.o.	t.o.	46489
14	208136	51693	41541	4606				74203
15	t.o.	80621	64788	6943				119579
16		121429	96808	9978				183907
17		180295	144835	14410				276432

determinant calculation [6]. Table 1 show the timing data taken to compute first derivatives of determinants of matrices of uni/bi-variate linear polynomial entries with random integer coefficients. The interpolation algorithm is distinctly superior to other algorithms as is expected, and the calculations complete within reasonable amounts of time even for fairly large matrices.

We now compare our method using ***ddet*** with the obvious method. Table 2 lists the time to compute the determinants of linear polynomial entry models with various number of variables (2, 3, 4 and 8), by the appropriate algorithm(s) to each case. It can be observed from the tables that reduction to uni/bi-variate cases by parameterization enabled by introducing the ***ddet*** operator, makes it possible to complete the calculation even with such large matrices of $N > 10$, whose determinant calculations are almost impossible with any algorithm if elements are multivariate. Even comparisons between bivariate (trivariate in case of multiplicity) determinant and univariate (bivariate) ***ddet*** calculations, assuming that we are able to perform substitutions for $U_i (i \neq k)$ in each matrix element before any determinant calculation as noted in the introduction, indicate the advantage of using our ***ddet*** operator: it can compute, at least, twice as fast as the obvious method, and furthermore we can expect the supression of the expression growth.

Table 2 Timing data in millisecond to compute the determinant of matrices of linear polynomial entries with random integer coefficients.

N	bivariate	trivariate		4-variate	8-variate
	INTP	NME	INTP	NME	NME
3	14	15	52	22	71
4	33	46	159	79	422
5	73	156	428	321	2387
6	147	524	1040	1192	11868
7	294	1735	2372	4161	53122
8	556	5529	5087	13656	216518
9	987	16302	10250	43503	t.o.
10	1691	46930	19504	131823	
11	2792	130505	35206	387877	
12	4530	t.o.	61347	t.o.	
13	7117		103395		
14	10644		165999		
15	15871		263061		
16	22866		t.o.		
17	32486				

6. Conclusions

In this paper, we have shown that elementwise substitutions for variables in a matrix is possible even for calculation of the first derivative of the determinant, based on the elementwise multiplicative property of determinant and the distribution law of differentiation. In order to formulate this computational process, we introduced a new operator ***ddet***, considering the differentiation operation after determinant expansion on a matrix as an operation on two matrices of the original elements and of their derivatives. The actual efficient algorithms for ***ddet*** are presented in this paper, in which recalculations of common subexpressions are eliminated by constructing the subexpressions from elements of two matrices systematically. In practice, our ***ddet*** operator is applicable to and very effective for obtaining symbolic solutions of a system of algebraic equations. Furthermore, we have set a matrix model of this particular problem, and have indicated the interpolation algorithm is best suited for the problem approved by the empirical studies.

Investigation of algebraic properties and the search for other application areas of our ***ddet*** operator are left for further study.

Acknowledgement

We would like to thank Mr. Taku Takeshima of Fujitsu Ltd. and Professor Shin Hitotsumatsu at Kyoto University (currently Tokyo Science University) for pointing out that the interpolation algorithm is best suited for the determinant calculations of our particular problem of solving a system of algebraic equations. Without it, we could not have reached the idea of introducing a new operator ***ddet***.

References

1. BAREISS, E. H. Sylvester's identity and multistep integer-preserving gaussian elimination, *Mathematics of Computation*, **22**, 103 (1968), 565-578.
2. GENTLEMAN, W. M. and JOHNSON, S. C. The evaluation of determinants by expansion by minors and the general problem of substitution, *Mathematics of Computation*, **28**, 125 (1974), 543-548.
3. GRISS, M. L. An efficient sparse minor expansion algorithm, *In Proceedings of ACM 76* (Houston, Texas, Oct. 20-22 1976), 429-434.
4. GRISS, M. L. Efficient expression evaluation in sparse minor expansion, using hashing and deferred evaluation, In Hawaii [5], 169-172.
5. *Proceedings of Hawaii International Conference on System Sciences* (Honolulu, Hawaii, Jan. 6-7 1977).
6. HOROWITZ, E. and SAHNI, S. On computing the exact determinant of matrices with polynomial entries, *J. ACM*, **22**, 1 (1975), 38-50.
7. KOBAYASHI, H., FUJISE, T. and FURUKAWA, A. Solving systems of algebraic equations by a general elimination method, *Journal of symbolic Computation*, **5**, 3 (June 1988), 303-320.
8. LIPSON, J. D. Chinese remainder and interpolation algorithms, *In Proceedings of Second Symposium on Symbolic and Algebraic Computation* (Los Angeles, Mar. 23-25 1971), 372-391.
9. MURAO, H., FUJISE, T. and KOBAYASHI, H. On factorizing symbolic *U*-resultant—application of ***ddet*** operator—. (submitted).
10. NORMAN, A. C. Symbolic and algebraic modes in REDUCE, *REDUCE Newsletter*, **3** (July 1978), 5-9.
11. SASAKI, T. and MURAO, H. Efficient Gaussian elimination method for symbolic determinants and linear systems, *ACM Trans. Math. Softw.*, **8** (1982), 277-289.
12. SMIT, J. The efficient calculation of symbolic determinant, *In Proceedings of 1976 ACM Symposium on Symbolic and Algebraic Computation* (Yorktown Heights, N.Y., Aug. 10-12 1976), 105-113.
13. WANG, P. S. On the expansion of symbolic determinants, In Hawaii [5], 173-175.

(Received August 8, 1991)