数理解析研究所講究録
第811巻 1992年 12-21

Use of Power-series Division in Multivariate Polynomial Arithmetic

M. Kolář *) and T. Sasaki **)

*) The Institute of Physical and Chemical Research (RIKEN)

Wako-shi, Saitama, 351-01 Japan

**) Institute of Mathematics, University of Tsukuba

Tsukuba-shi, Ibaraki, 305 Japan

1. Introduction

This paper investigates the computation of quotient of two multivariate polynomials by neglecting higher degree terms of dididend and divisor. If a multivariate polynomial is exactly divisible by another then the power-series division allows us to calculate the quotient, and in the power-series division we can neglect the unnecessary higher degree terms. However, so long as the authors know, no literature tells us which terms are unnecessary when the divident and divisor are multivariate polynomials. So, in the first half of this paper, we clarify it.

If the "size" of quotient is much smaller than the sizes of divident and divisor, the power-series method with the higher degree terms discarded will be more efficient than the conventional division method. In order to see how the power-series division method is useful in practical calculations, we test this method in the fraction-free Gaussian elim-

ination algorithm in the second half of this paper.

2. Multivariate power-series division

Number of terms of two exactly divisible polynomials that are essential for the power series division (significant terms) is different when the division is done with respect to the total degree and when it is done recursively with respect to individual variables. In the second case it may depend on the order of the variables. The first case is treated in Lemma 2 of Ref. 1. The case of the recursive division will be discussed here.

Definition1

Let $\deg(P,u)$ be the highest power of the variable u occurring in the polynomial P.

Let $\mathrm{lp}(P,u)$ be the lowest power of the variable u occurring in the polynomial P.

Let $\mathrm{lc}(P,u)$ be the coefficient of the lowest power of the variable u occurring in P. $\diamond$

Example: $P = x^3y^4z^2 + 2xy^2z^2 + z^3 + z^2$. Then $\deg(P,x) = 3$, $\mathrm{lp}(P,z) = 2$, $\mathrm{lc}(P,z) = x^3y^4 + 2xy^2 + 1$, $\mathrm{lp}(\mathrm{lc}(P,z),x) = 0$, etc. $\diamond$

Theorem1 Let A and B be polynomials in n variables $u_0, u_1, \ldots, u_{n-1}$. Let A be exactly divisible by B. Let the power series division A/B be done recursively in the above order of variables (GAL: ORDER u_0, u_1, ..., u_{n-1}). Let $B_0 = B$ and $B_i = \mathrm{lc}(B_{i-1}, u_{i-1})$; $i = 1,\ldots,n-1$ (B_i is a polynomial in $u_i, \ldots, u_{n-1}$). Then to calculate the ratio $C = A/B$ by means of the recursive power-series division, it is sufficient to know only those terms T of A and B for which

$$\deg(T,u_i) \leq \deg(A,u_i) - \deg(B,u_i) + \mathrm{lp}(B_i,u_i)\,,\ \subset = 0,\ldots,n-1. \qquad (1)$$

In other words, in GAL one can introduce the following declarations:

$$\text{DECL POWSER} : u_i <= \deg(A,u_i) - \deg(B,u_i) + \mathrm{lp}(B_i,u_i) \qquad (2)$$

Proof: Let us write

$$A = \sum_{n=N_1}^{N_2} a_n u_0^n, \quad B = \sum_{m=M_1}^{M_2} b_m u_0^m, \quad C = \sum_{k=N_1-M_1}^{N_2-M_2} c_k u_0^k.$$

Here a_n, b_m and c_k are polynomials in $u_1, \ldots, u_{n-1}$. Obviously,

$$c_{N_1-M_1+\kappa} = \frac{a_{N_1+\kappa} - \sum_{j=1}^{J_\kappa} b_{M_1+j}\, c_{N_1-M_1+\kappa-j}}{b_{M_1}}, \quad \kappa = 0, \ldots, \Delta, \tag{3}$$

where $J_\kappa = \min(\kappa, M_2 - M_1)$ and $\Delta = N_2 - M_2 - (N_1 - M_1)$. To determine all the $\Delta + 1$ coefficients c_k, the knowledge of a_{N_1}, a_{N_1+1}, ..., $a_{N_2-M_2+M_1}$ and b_{M_1}, b_{M_1+1}, ..., $b_{M_1+\min(\Delta, M_2-M_1)}$ is sufficient according to Eq. (3). All other terms of A and B can be truncated. Therefore, the maximum power of u_0 that must be kept in A and B is $m_A = N_2 - M_2 + M_1$ and $m_B = M_1 + \min(\Delta, M_2 - M_1)$, respectively. It is easy to check that always $\max(m_A, m_B) = m_A$. Therefore, it is sufficient to truncate in A and B all terms with the power of u_0 larger than

$$m_A = \deg(A, u_0) - \deg(B, u_0) + \mathrm{lp}(B, u_0). \tag{4}$$

Using this common truncation cutoff for both A and B may mean that we keep some redundant terms in B but it is not feasible to have separate truncation cutoffs for A and B in a computer algebra system. Thus we have proved Eq. (1) for the leading variable u_0.

To prove it for all other variables, we will follow the recursive division. Equation (3) represents again polynomial division, this time with leading variable u_1. Let us call the divisions in Eq. (3) the 1st level divisions while the original division $C = A/B$ is the 0th level division. Generally, during the $(i+1)$th level division with the leading variable u_{i+1}, one calculates certain coefficients $\gamma_{e_0 e_1 \cdots e_i}$ that contribute to C in the following way:

$$C = \gamma_{e_0 e_1 \cdots e_i}(u_{i+1}, u_{i+2}, \ldots, u_{n-1})\, u_0^{e_0} u_1^{e_1} \cdots u_i^{e_i} + \cdots. \tag{5}$$

Coefficients γ are obtained by divisions similar to that in Eq. (3):

$$\gamma_{e_0 e_1 \cdots e_i}(u_{i+1}, u_{i+2}, \ldots, u_{n-1}) = \frac{\alpha(u_{i+1}, u_{i+2}, \ldots, u_{n-1})}{B_{i+1}}. \tag{6}$$

In Eq. (6), the leading variable is u_{i+1}, α plays the same role as A in the 0th level division, B_{i+1} plays the role of $B \equiv B_0$, and $\gamma_{e_0 e_1 \cdots e_i}$ that of C. Therefore, we can apply the result of Eq. (4), which means that in Eq. (6) one can truncate all terms of α and B_{i+1} with the power of u_{i+1} larger than

$$m_{e_0 e_1 \cdots e_i} = \deg(\alpha, u_{i+1}) - \deg(B_{i+1}, u_{i+1}) + \mathrm{lp}(B_{i+1}, u_{i+1}), \tag{7}$$

From Eq. (6) we have

$$\deg(\alpha, u_{i+1}) - \deg(B_{i+1}, u_{i+1}) = \deg(\gamma_{e_0 e_1 \cdots e_i}, u_{i+1}). \tag{8}$$

From Eq. (5) we have

$$\deg(\gamma_{e_0 e_1 \cdots e_i}, u_{i+1}) \leq \deg(C, u_{i+1}). \tag{9}$$

Obviously,

$$\deg(C, u_{i+1}) = \deg(A, u_{i+1}) - \deg(B, u_{i+1}). \tag{10}$$

Combining Eqs. (7-10) gives

$$m_{e_0 e_1 \cdots e_i} \leq \deg(A, u_{i+1}) - \deg(B, u_{i+1}) + \mathrm{lp}(B_{i+1}, u_{i+1}). \tag{11}$$

To obtain correct results for all $(i+1)$th level divisions, we have to make the cutoff at the maximum of all $m_{e_0 e_1 \cdots e_i}$ which is just the right-hand side of Eq. (11) because the equality in Eq. (9) must hold for at least one combination of exponents $e_0 e_1 \cdots e_i$. This completes the proof. $\diamond$

3. Application to Determinant Calculation

Let M be the following $N \times N$ matrix

$$M = \begin{pmatrix} M_{1,1} & M_{1,2} & \cdots & M_{1,N} \\ M_{2,1} & M_{2,2} & \cdots & M_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ M_{N,1} & M_{N,2} & \cdots & M_{N,N} \end{pmatrix}.$$

We will calculate the determinant $D = |M|$ by the fraction-free Gaussian elimination method. For $k = 2, \cdots, N-1$ and $k \leq i, j \leq N$, let

$$D(i,j)^{(k)} = \begin{vmatrix} M_{1,1} & \cdots & M_{1,k-1} & M_{1,j} \\ \vdots & \ddots & \vdots & \vdots \\ M_{k-1,1} & \cdots & M_{k-1,k-1} & M_{k-1,j} \\ M_{i,1} & \cdots & M_{i,k-1} & M_{i,j} \end{vmatrix}.$$

Then, in the fraction-free Gaussian elimination algorithm applied for calculating the determinant D, one uses the following recursion formula:

$$D(i,j)^{(k+1)} = \left(D(k,k)^{(k)} D(i,j)^{(k)} - D(i,k)^{(k)} D(k,j)^{(k)}\right) / D(k-1,k-1)^{(k-1)}.$$

This is a situation when we know in advance that two polynomials are exactly divisible. In addition to that, the numerator is being calculated just before the division. Therefore, in GAL one can declare the truncation rules (2) according to Theorem 1, which will result in automatic truncation of all insignificant terms during the calculation of the numerator and automatic use of the power series division. This should substantially decrease, at least for some types of matrices, the intermediate expression swelling, and as a consequence also decrease the execution time. In this section we compare the CPU times necessary for the calculation of the determinants of various matrices without and with the use of the truncation.

Procedure DETG uses the plain fraction-free Gaussian elimination algorithm without any truncation. DETG1 and DETG1E are two variations of the same algorithm that

implements the declarations of Eq. (2) for all variables occurring in the matrix. They differ only in how $\deg(D(i,j)^{(k)}, u_l)$ is calculated. DETG1 uses the procedure VDEG() separately for each u_l. DETG1E calculates the degrees with respect to all variables simultaneously using the procedure EMAXMIN(). The upper bound of the degree of the numerator to be used in Eq. (2) is then

$$\max\left(\deg(D(k,k)^{(k)}, u_l) + \deg(M(i,j)^{(k)}, u_l),\ \deg(D(i,k)^{(k)}, u_l) + \deg(D(k,j)^{(k)}, u_l)\right).$$

As will be seen below, DETG1E gives shorter execution times than DETG1 when the number of variables occurring in the matrix is large (at least 7 or 8). Finally DETG2 introduces an auxiliary total degree variable T by means of the substitution

$$u_i \mapsto u_i T\ ,$$

T is made the leading variable of the 0th level division and the truncation is done with respect to T only.

Execution times in milliseconds for various types of matrices are in the following tables:

Table I. Matrices from the file MATRX1.DATA: "Random" matrices with relatively large differences in the number of variables occurring in individual matrix elements and in their complexity. N is the order of the matrix and n is the total number of variables occurring in all matrix elements.

N	n	*DETG*	*DETG*1	*DETG*1*E*	*DETG*2
3	5	170	197	218	234
4	6	2289	2085	2081	3295
5	6	13952	11981	12408	43403
6	6	11377	9714	10056	25701
7	5	42066	42141	43180	store jam

Table II. Matrices from the file M2.DATA: "Special" matrices: $M_{1,1}$ is given by a long expression containing all 8 variables while all other elements contain at most one variable of power 1 with the exception of a single element equal to the square of a variable. N and n are as in Table I.

N	n	$DETG$	$DETG1$	$DETG1E$	$DETG2$
6	8	16620	8490	8242	6233
7	8	store jam	19527	19208	14424

Table III. Matrices from the file M3.DATA: "Regular" matrices given by the formula $M_{ij} = x_i\, x_j + (1 + x_i^2)\, \delta_{ij}$. N and n are as in Table I.

N	n	$DETG$	$DETG1$	$DETG1E$	$DETG2$
3	3	16	44	54	32
4	4	57	147	179	81
5	5	190	422	484	209
6	6	611	1075	1171	559
7	7	1937	2553	2563	1546
8	8	6266	5739	5537	4708
9	9	21036	13107	12304	15808
10	10	72692	30665	28286	store jam
11	11	store jam	74586	68817	store jam

Table IV. Modified matrices from the file M3.DATA: "Regular" matrices given by the

formula $M_{ij} = x_i x_j + \delta_{ij}$. N and n are as in Table I.

N	n	$DETG$	$DETG1$	$DETG1E$	$DETG2$
7	7	221	1232	1274	294
8	8	364	2136	2143	437
9	9	579	3577	3498	636
10	10	851	5527	5410	881
11	11	1226	8392	8052	1223
12	12	1763	12264	11558	1625
13	13	1859	15219	13965	1830
14	14	3106	23208	21746	2612
15	15	4136	31554	28979	3224

Generating the truncation rules (2) represents some overhead processing that can in cases when few truncations occur cancel or even exceed the gains obtained by truncation. It seems that for general matrices such as those of Table I, the efficiency of procedures DETG1 or DETG1E is better or at least comparable with that of DETG. In some special cases, DETG2 can be even more efficient than DETG1/DETG1E, but on the other hand it can fail in some cases when DETG1/DETG1E is rather efficient. Although more calculations are needed to devise a general strategy, on the basis of our preliminary results it seems that one could proceed as follow: use DETG for matrices of small order and with simple expressions with very few variables for *all* elements. Use DETG1 in other cases when the number of variables is less than 8, otherwise use DETG1E. If DETG1/DETG1E fails try DETG2. If determinants of a large number of matrices of the same class are to be calculated, try all the above procedures on a representative sample of the matrices and chose the one with the best results.

The matrices of Table IV represent a rather degenerate case. Although they differ only in the value of a coefficient in their diagonal elements from the matrices of Table III, the resulting determinants differ considerably. The diagonal elements of the matrices of Table III are equal to $1 + 2x_i^2$ while those of the matrices of Table IV are equal to $1 + x_i^2$. The determinants of the first class of matrices are given by expressions several pages long, while the determinants of the matrices of Table IV are equal to the simple expression $1 + \sum_{i=1}^{n} x_i^2$. Moreover the off-diagonal elements of these matrices do not change at all during the Gaussian elimination. The diagonal elements at the end of the Gaussian elimination process are $M_{kk} = 1 + \sum_{i=1}^{k} x_i^2$. Because the elements remain so simple during the whole process, there is almost no truncation. That explains the times of Table IV.

4. Conclusion

Tables I, III and IV show that the power-series division method is not useful for the fraction-free Gaussian algorithm. In the case of determinant calculation, on an average, the quotient is a larger-sezed polynomial than the divisor. We think this is the reason why the power-series division is not useful in our test. If the divisor is a much larger polynomial than the quotient, we will obtain a much better result, as the result in Table II shows. Therefore, when we use the power-series division method, we must select the problems carefully. In fact, we already know that, in the case of GCD calculation, the method is very useful as ref. 1 shows.

Acknowledgments

M.K. is grateful to the Japanese Science and Technology Agency for financial support, and to the Information Science Laboratory of the RIKEN Institute for hospitality.

References

1. T. Sasaki and M. Suzuki, *Three New Algorithms for Multivariate Polynomial GCD.* To be published in J. Symbol. Comput.