Journal of Computational Mathematics, Vol.10, No.4, 1992, 296–300.

THE RANK–k UPDATING ALGORITHM FOR THE EXACT INVERSION OF MATRICES WITH INTEGER ELEMENTS*

Deng Jian-xin
(*Computing Center, Academia Sinica, Beijing, China*)

Abstract

In this paper, the numerical solution of the matrix problems over a ring of integers is discussed. The rank-k updating algorithm for the exact inversion of a matrix is proposed. This algorithm is generally more effective than Jordan elimination. The common divisor of the numbers involved is reduced to avoid over-swelling of intermediate numbers.

§1. Introduction

Solving such problems as arising in number theory, graph theory, coding theory, statistics, linear programming and circuit theory require computing the exact inversion of a matrix with integer elements. The exact solution of a problem of a matrix also plays an important role in numerical experiments. For example, when matrices are ill-conditioned, the integral methods are more interesting. What is most difficult in integer computation is the rapid increase of the numbers involved when the usual algorithms are applied. So, first of all, an integer algorithm must be able to restrain intermediate numbers from swelling. Many algorithms have been developed[2–6], such as

(a) Elementary transformations method;

(b) Division-free Gaussian elimination;

(c) One-step Jordan elimination;

(d) Multistep Gaussian elimination;

(e) Congruence method;

(f) Determinant and Gramer's rule.

In this paper, a rank-k updating algorithm for the exact inverse matrix and the exact solution of a system of linear equations is proposed. This algorithm is generally more effective than the Jordan elimination. The common divisors of the numbers involved are reduced to restrain intermediate numbers from swelling.

§2. The Rank-k updating algorithm

Suppose that A is a nonsingular n by n integer matrix and u, v are n by k rank-k integer matrices. Then uv^t is a rank-k matrix and $A + uv^t$ is a rank-k updating matrix.

* Received December 25, 1987.

It is easy to prove that the matrix $A + uv^t$ satisfies the following identical relation:

$$\det\ (A + UV^t) = \det(A)\det(I + v^tA^{-1}u). \tag{2.1}$$

The inversion of matrix $A + uv^t$ can be represented by the following formula if the determinant of $A + uv^t$ is non-zero:

$$(A + uv^t)^{-1} = A^{-1} - A^{-1}u(I + v^tA^{-1}u)^{-1}v^tA^{-1}. \tag{2.2}$$

Define the following notation:

$$n = m * k, \quad A = (a_1, a_2, \cdots, a_n)^t, \quad A_0 = I,$$
$$U_i = (e_{ik+1}, e_{ik+2}, \cdots, e_{(i+1)k}), \quad \widetilde{V}_i = (a_{ik+1}, a_{ik+2}, \cdots, a_{(i+1)k}), \quad V_i = \widetilde{V}_i - U_i,$$

where I is an identity matrix, with i-th column as e_i. With this notation, the following representations are obtained:

$$A_i = A_{i-1} + u_{i-1}v_{i-1}^t, \quad A = A_m, \quad i = 1, 2, \cdots, m. \tag{2.3}$$

The exact det (A) and det $(A)A^{-1}$ are computed by determining the sequence of det (A_i) and det $(A_i)A_i^{-1}$. Formula (2.2) shows that

$$A_1^{-1} = (I + u_0v_0^t)^{-1} = I - u_0(I + v_0^tu_0)^{-1}v_0^t. \tag{2.4}$$

Let

$$d_1 = det(I + v_0^tu_0), \quad w_1 = d_1(I + v_0^tu_0)^{-1}. \tag{2.5}$$

From equation(2.1), $d_1 = \det\ (A_1)$. If d_1 is non-zero, the relation (2.4) holds. It follows that

$$d_1A_1^{-1} = d_1I - u_0w_1v_0^t. \tag{2.6}$$

The matrix $d_1A_1^{-1}$ is an adjoint matrix of A_1, so the elements of the matrix are integers. Generally, there is no common divisor in the matrix, and the same is true with matrix w_1. From relation (2.2),

$$\begin{aligned} A_2^{-1} &= (A_1 + u_1v_1^t)^{-1} = A_1^{-1} - A_1^{-1}u_1(I + v_1^tA_1^{-1}u_1)^{-1}v_1^tA_1^{-1} \\ &= A_1^{-1} - A_1^{-1}u_1d_1(d_1I + v_1^td_1A_1^{-1}u_1)^{-1}v_1A_1^{-1} \end{aligned} \tag{2.7}$$

Let $d_2 = \det\ (A_2)$. Apply relation (2.1) again to obtain

$$d_2 = \det\ (d_1I + v_1^td_1A_1^{-1}u_1). \tag{2.8}$$

If w_2 is defined as

$$w_2 = d_2(d_1I + v_1^td_1A_1^{-1}u_1)^{-1}, \tag{2.9}$$

then equation (2.7) becomes

$$d_2 d_1 A_2^{-1} = d_2 d_1 A_1^{-1} - d_1 A_1^{-1} u_1 w_2 v_1^t d_1 A_1^{-1}. \tag{2.10}$$

Since $d_2 A_2^{-1}$ is an adjoint matrix of A_2, the elements of matrix $d_2 d_1 A_2^{-1}$ must have a common divisor d_1.

Generally,

$$\begin{aligned} A_{i+1}^{-1} &= (A_i + u_i v_i^t)^{-1} = A_i^{-1} - A_i^{-1} u_i (I + v_i^t A_i^{-1} u_i)^{-1} v_i^t A_i^{-1} \\ &= A_i^{-1} - A_i^{-1} u_i d_i (d_i I + v_i^t d_i A_i^{-1} u_i)^{-1} v_i^t A_i^{-1} \end{aligned} \tag{2.11}$$

If $d_{i+1} \neq 0$, equation (2.11) holds. The following matrix recurrence relations can be found:

$$d_{i+1} d_i A_{i+1}^{-1} = d_{i+1} d_i A_i^{-1} - d_i A_i^{-1} u_i w_{i+1} v_i^t d_i A_i^{-1}. \tag{2.13}$$

Reducing the common divisor, we obtain an integer matrix $d_{i+1} A_{i+1}^{-1}$. There is no breakdown in the recursive process, if

$$d_j \neq 0, \quad j = 1, 2, \cdots, m. \tag{2.14}$$

The exact det (A) equals det (A_m) and the exact det $(A)A^{-1}$ equals det $(A)A_m^{-1}$ finally. Obviously, there exists a permutation matrix P so that all leading principal determinants of matrix PA are non-zero. Therefore, (2.14) can always be satisfied. This condition is necessary for other eliminations too.

§3. The Rank-1 updating algorithm

In the simplest case $k = 1$, the rank-1 updating algorithm can be described in detail as follows.

Rank-1 updating algorithm.

(a) Let $d_0 = 1$, $A_0 = I$.

(b) For $i = 1, 2, \cdots, n$,

$$d_i = a_i^t d_{i-1} A_{i-1}^{-1} e_i,$$

interchange the i-th row and the $(i+j)$-th row, if $d_i = 0$, for $j = 1, 2, \cdots, n-i$, if $d_i = 0$. Stop (A is singular).

$$d_i A_i^{-1} = (d_i d_{i-1} A_{i-1}^{-1} - d_{i-1} A_{i-1}^{-1} e_i (A_i^t - e_i^t) d_{i-1} A_{i-1}^{-1}) / d_{i-1} \tag{3.1}$$

(c) According to the changing of the rows in (b), interchange the corresponding columns of $d_n A_n^{-1}$.

(d) Let det $(A) =$ det (A_n), and det $(A)A^{-1} =$ det $(A_n)A_n^{-1}$. Stop.

§4. Discussion

The Rank-k algorithm can considerably reduce the amount of arithmetic for some matrices whose non-zero elements are distributed specially. Particularly, for the Rank-1 algorithm, the pattern of non-zero elements in A_{i-1} and $d_{i-1}A_{i-1}^{-1}$ appear as

$$\begin{array}{c|c}
\begin{matrix} * & * & \bullet & \bullet & \bullet & * \\ * & * & \bullet & \bullet & \bullet & * \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ * & * & \bullet & \bullet & \bullet & * \end{matrix} & \begin{matrix} * & * & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & * \\ * & * & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & * \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & * \\ * & * & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & * \end{matrix} \\ \hline
 & \begin{matrix} 1 & & & & & & & & \\ & 1 & & & & & & & \\ & & \bullet & & & & & & \\ & & & \bullet & & & & & \\ & & & & \bullet & & & & \\ & & & & & \bullet & & & \\ & & & & & & \bullet & & \\ & & & & & & & \bullet & \\ & & & & & & & & 1 \end{matrix}
\end{array}
\begin{matrix} i-1 \\ \\ n-i+1 \end{matrix}
\qquad \text{(3.2)}$$

$$i-1 \qquad\qquad n-i+1$$

Since the elements from $i+1$ to n in the column $d_{i-1}A_{i-1}^{-1}e_i$ are all zero, only the first i rows of the matrix should be updated. If $j < i$, the first $i-1$ elements are non-zero only in the column $d_{i-1}A_{i-1}^{-1}e_j$. If $j \geq i$, the first $i-1$ elements and j-th element are non-zero only in the column. So the main arithmetic is only concerned with first $i-1$ elements in $(a_i^t - e_i^t)d_{i-1}A_{i-1}^{-1}$. The number of multiplications which transform $d_iA_i^{-1}$ into $d_{i+1}A_{i+1}^{-1}$ is about $3 * i * n$, and the number of divisions is $i * n$. If the number of divisions was the same as of multiplications, the amount of operation in the algorithm (3.1) would be $NR \sim (4 * n^3 + 6 * n)/2$. In the Jordan elimination algorithm, the corresponding amount is $NJ \sim (9 * n^3 - 12 * n)/2$. The effect of the algorithm is the same as Jordan elimination on the 'intermediate word swell'. So the rank-1 updating algorithm is more efficient than the Jordan elimination. Furthermore, the Rank-k algorithm is more efficient than the Rank-1 algorithm for special types of matrix with special distribution of non-zero elements.

§5. FORTRAN Routine

A FORTRAN routine LRANK is designed by the Scientific Software Methodology and Technology Deptartment, Computing Center, Academia Sinica. The routine implements the rank-1 updating algorithm (3.1) using MP-system[1]. It can be used in FORTRAN compiler environment of a variety of computers and be applied to compute exact det $(A)A^{-1}$, where A is an n by n-matrix with any word-length integer elements. In practice, the scale of the computing problem is limited by the capacity of the computer.

References

[1] Computing Center, Academia Sinica, The high precision arithmetic system, 1985.

[2] E.H. Bareiss, Computational solutions of matrix problem over an integral domain, *J. Inst. Math. Appl.*, **10** (1972), 68–104.

[3] G.N.Lance, Numerical Methods for High Speed Computers, Iliffe & Sons, London, 1960.

[4] S.Cabay and T.P.L.Lam, Congruence techniques for the exact solution of integer systems of linear equations, *ACM Transactions on Mathematical Software*, **3** (1977), 386–397.

[5] S.Cahay and T.P.L.Lam, Algorithm 522 Esolve, Congruence techniques for the exact solution of integer systems of linear equations, *ACM Transactions on Mathematical Software*, **3** (1977), 404–407.

[6] Jörn Springer, Exact solution of general integer systems of linear equations, *ACM Transactions on Mathematical Software*, **12** (1986), 51–61.