Computer Algorithm for the Recursive Method of Calculating Large Numerator Relationship Matrices

G.F.S. HUDSON, R. L. QUAAS, and L. D. VAN VLECK
Department of Animal Science
Cornell University
Ithaca, NY 14853

ABSTRACT

A method of storing and retrieving nonzero elements of a large, sparse symmetric matrix using only internal computer storage is presented. An algorithm for applying the recursive or tabular method of calculating the numerator relationship matrix of a large group of animals is given. The algorithm is applicable to inbred and noninbred populations and can be used on sire/dam or sire/maternal grandsire pedigrees.

INTRODUCTION

Wright's (9) coefficients of relationship and inbreeding are calculated most easily by the well-known recursive or tabular method attributed to J. L. Lush by Emik and Terrill (1). Henderson (4) and Quaas (6) have developed methods for computing the inverse of the numerator relationship matrix required for best linear unbiased predictions (BLUP) of breeding values (3) without calculating and inverting the matrix directly. However, knowledge of relationships among a group of animals is useful for examining the effect of, and preventing, inbreeding. Furthermore, the numerator relationship matrix, A say, is required if relationships among animals are incorporated into variance component estimation models (8).

This paper describes an algorithm for storing and computing A for large groups of animals. The method is explained by the hypothetical example of Table 1 in which animals have been numbered consecutively in decreasing age order. The corresponding relationship matrix is in Table 2.

STORING NONZERO ELEMENTS OF A

Suppose A has order N^2 and contains n nonzero elements in the upper half (i.e., on and above the diagonal). The ijth element, a_{ij}, is the numerator relationship between animals i and j. The entire upper half could be stored in a single array of dimension N(N + 1)/2, but computer storage space becomes limiting when N is more than a few hundred. The method presented requires storing only nonzero elements as half-word integers in an array of dimension n. Two half-word integer arrays, of dimension n and N, store information necessary to recover a given element of the original matrix A. On the IBM 370 computer a full-word (or long) integer requires 32 bits of memory whereas a half-word (or short) integer uses only 16 bits of memory. The capability to use half-word integers is dependent on the computer available, but the principles are applied easily to any system.

To reduce storage requirements and increase the order of the matrix that can be calculated, relationship coefficients are expressed as integers by multiplication by a large power of 2, e.g., 2^{14}. The integers can be stored in a half-word integer array rather than the full-word floating-point array necessary to store a decimal fraction. For example, relationships 1/2, 3/8, 5/16 are stored as 8192, 6144, and 5120. Other coefficients are expressed in a similar manner. For clarity in this paper, relationships are left in more familiar decimal form.

The storage method is similar to Scheme II of (7). Nonzero elements of the upper-half of A are stored in rows: in one array, COEFF, of dimension n (Table 3). The column subscripts of off-diagonal elements of A are stored in a half-word integer array of dimension n, COL. For example, the 11th location in COEFF contains an element of the 5th column of A. Locations of diagonal elements in COEFF are stored in a half-word integer array, ROW, of dimension N. For example, the location of the 4th diagonal in COEFF is stored in the 4th entry of ROW. Table 3 shows the contents of the arrays after all calculations have been

Received February 26, 1982.

TABLE 1. Example pedigree.

Animal	Sire	Dam
1	...[a]	...
2	...	...
3	...	...
4	1	3
5	2	3
6	2	...
7	2	6

[a]Missing information.

completed. At the beginning of the computations n is unknown, so the necessary length of COEFF and COL has to be estimated.

Recovery of any particular element of A is accomplished by searching for the column subscript in the interval of COL defined by the diagonal elements of the current and next rows. Within each row, the elements of COL are in strictly increasing order; thus, a bisection search can be used to determine the location or absence of the required column subscript. Efficient search algorithms are in (5), but an outline of a bisection search follows. The interval of COL that contains the column subscripts of the appropriate row is bisected to find the approximate midpoint of the interval. The entry of COL at the midpoint is compared with the search argument, and the upper or lower half of the interval is taken as a new interval depending on whether the midpoint is less than or greater than the search argument. The process continues until the required entry is found at the midpoint or until the interval collapses in which case the search argument is not in the original interval. If an element below the diagonal is required, e.g., $a_{i,j}$ with $i > j$, the subscripts must be interchanged because only the upper half of A is stored. Recovery of diagonal elements requires only the location in ROW, e.g., $a_{3,3}$ is known to be stored in COEFF (7) as ROW (3) contains 7.

PROCEDURE FOR CALCULATING NONZERO ELEMENTS OF A

Animals must be identified in decreasing age order with sequential numbers from 1 to N, so that 1 identifies the oldest animal and N the youngest. Sire and dam identification for each animal are stored in two half-word integer arrays each of dimension N. A zero is stored if parental information is missing.

For each row of A, the diagonal element is calculated first, followed by all off-diagonals to the right of the diagonal. In the last row there are no off-diagonals, but the diagonal has to be computed for completeness of the matrix. Explanation of the algorithm is simplier if we assume that the first $i - 1$ rows have already been computed and stored in the manner described in the previous section.

The diagonal element of the ith row, a_{ii}, is $1 + f_i$, where $f_i = 1/2\ a_{jk}$ is the inbreeding coefficient of the ith animal and j and k are the sire and dam of the ith animal (not necessarily respectively), with $k > j$ for the search procedure. If either sire or dam identification is missing or if $a_{jk} = 0$, then $a_{ii} = 1$, except as indicated below where only sire and maternal grandsire are identified. The a_{ii} is stored in the next available entry of COEFF, and its position stored in the ith entry of ROW. (A counter is needed to keep track of the number of filled entries in COL and COEFF.)

TABLE 2. Numerator relationship matrix for the example of Table 1.

Animal	1	2	3	4	5	6	7
1	1	0	0	1/2	0	0	0
2	0	1	0	0	1/2	1/2	3/4
3	0	0	1	1/2	1/2	0	0
4	1/2	0	1/2	1	1/4	0	0
5	0	1/2	1/2	1/4	1	1/4	3/8
6	0	1/2	0	0	1/4	1	3/4
7	0	3/4	0	0	3/8	3/4	1+1/4