Genet Sel Evol (1992) 24, 305-313

Original article

Computing inbreeding coefficients in large populations

THE Meuwissen *, Z Luo

Institute of Animal Physiology and Genetics Research, Edinburgh Research Station, Roslin EH25 9PS, UK

(Received 21 October 1991; accepted 15 April 1992)

Summary – An algorithm for computing inbreeding coefficients in large populations is presented. It is especially useful in large populations because of the small size of the memory required, which is linear with population size, and its speed, if the number of generations involved is not too large, *ie* not larger than about 12. The method is compared with 2 other methods for computational speed and memory requirement. The presented algorithm is suited for situations where the inbreeding coefficients for a few new animals are to be computed given that their ancestor's inbreeding coefficients were calculated previously.

inbreeding coefficient / algorithm

Résumé – Le calcul des coefficients de consanguinité dans de grandes populations. *On présente un algorithme de calcul des coefficients de consanguinité pour de grandes populations. Il est particulièrement adapté à ces populations à cause du faible volume de mémoire d'ordinateur requis (il dépend linéairement de la taille de la population) et de la rapidité de calcul quand le nombre de générations impliquées n'est pas trop élevé (c'est-à-dire inférieur à environ 12). La méthode est comparée à 2 autres méthodes du point de vue de la vitesse de calcul et de la mémoire requise. Le présent algorithme est également adapté aux situations de mise à jour où les coefficients de consanguinité correspondant à un faible nombre d'animaux nouveaux doit être calculé en tirant parti des coefficients calculés pour les anciens animaux.*

coefficient de consanguinité / algorithme

* On leave from the Schoonoord Research Institute for Animal Production, PO Box 501, 3700 AM Zeist, The Netherlands

INTRODUCTION

Many countries are implementing or have implemented evaluation methods for national herds based upon animal models. If these account for inbreeding, calculation of inbreeding coefficients in large populations is required. Henderson (1976) and Quaas (1976) implicitly present methods for the calculation of inbreeding coefficients, when they propose algorithms for the calculation of the inverse of the additive relationship matrix. Henderson's method requires storage of a large matrix. Quaas avoids this: memory requirement is linear with N and computation time is proportional to N^2, where N is the size of the data set. In Quaas' method computer time becomes a limiting factor with increasing N. With this algorithm, no use is made of known inbreeding coefficients, and hence all calculations have to be repeated whenever coefficients are required for a new batch of animals. Golden *et al* (1991) present an algorithm based on Quaas' method using sparse programming techniques.

Tier (1990) presents a fast method for the calculation of inbreeding coefficients. Using sparse programming techniques, his algorithm first determines which elements in the additive genetic relationship matrix A are required and then calculates them. The memory required is a small proportion of N^2: about 0.9%. For large populations, the physical memory of the computer is still likely to be too small and use of disk memory is required, slowing this algorithm considerably. Known inbreeding coefficients are not recalculated.

The aim of this paper is to present a method for the calculation of inbreeding coefficients in large populations, which is fast, requires memory proportional to N and does not recompute known inbreeding coefficients. The method presented is compared to Golden *et al*'s (1991) implementation of Quaas' (1976) method and Tier's (1990) method for speed and memory required.

METHOD

The method is based on the decomposition of the additive genetic relationship matrix $\mathbf{A}$, as described by Henderson (1976): $\mathbf{A} = \mathbf{LDL}'$, where $\mathbf{L}$ is a lower triangular matrix containing the fraction of the genes that animals derive from their ancestors, and $\mathbf{D}$ is a diagonal matrix containing the within family additive genetic variances of animals. The animals are ordered such that parents precede offspring. From the decomposition, it follows that (Quaas, 1976):

$$A_{ii} = \sum_{j=1}^{i} L_{ij}^2 D_{jj}, \qquad [1]$$

where A_{ii} is the ith diagonal element of $\mathbf{A}$, which equals the inbreeding coefficient of animal i plus 1. Quaas (1976) computes the elements of $\mathbf{L}$ quickly and one column at a time by the following recursive algorithm:

For $j = 1$ to N (all columns of $\mathbf{L}$)
  $L_{jj} = 1$

For $i = j + 1$ to N (all elements below the diagonal element)

$L_{ij} = (L_{s_i j} + L_{d_i j})/2$ when both parents s_i and d_i of i are known, or
$= L_{k_i j}/2$ when only one parent k_i of i is known, or
$= 0$ when both parents are unknown, for $i < j$.

The elements of $\mathbf{D}$ are calculated as:

$D_{jj} = 1$ when both parents are unknown, or
$= 0.75 - F_{k_j}/4$ when only one parent k_j of j is known, or
$= 0.5 - (F_{s_j} + F_{d_j})/4$ when both parents s_j and d_j of j are known, where $\mathbf{F}_j$ denotes the inbreeding coefficient of animal j. After computing the elements of the jth column of $\mathbf{L}$, they are squared and multiplied by D_{jj}. The resulting vector is added to a working vector. When this procedure is followed for every column of $\mathbf{L}$, the working vector contains the A_{jj}-values. This algorithm requires $N(N+1)/2$ operations. Golden *et al* (1991) made a list of the offspring of all the animals i. Because element L_{ij} is non-zero only if i is a descendant of j, the operations within each column are performed only for the descendants of j. Because the elements of the columns of $\mathbf{L}$ are not stored they have to be recalculated when a new batch of animals becomes available.

In the present algorithm, the elements of $\mathbf{L}$ are computed row by row, which overcomes the problem of recalculating elements of $\mathbf{L}$ when a new batch of animals has to be evaluated. A row i of $\mathbf{L}$ gives the fraction of the genes that animal i derives from its ancestors. Hence, $L_{is_i} = L_{id_i} = 0.5$, where s_i and d_i are the sire and dam of i, respectively. Each row of $\mathbf{L}$ can be built by proceeding up the pedigree adding half the "contribution" of the current animal to each of its parents. The fraction of the genes derived from an ancestor is:

$$L_{ij} = \sum_{k \varepsilon \mathbf{P}_j} 1/2 L_{ik} = \sum_{k \varepsilon \mathbf{ANC}_i \cap \mathbf{P}_j} 1/2 L_{ik}, \qquad [2]$$

where $\mathbf{P}_j$ is the set of identification numbers of the progeny of j and $\mathbf{ANC}_i$ is the set of identification numbers of the ancestors of i, including animal i itself. The latter identity in [2] is because $L_{ik} = 0$, if k is not an ancestor of i or not equal to i.

This leads to the following algorithm for computing $\mathbf{L}$, row by row. As each row is determined, the contributions to the elements of A_{ii} are accumulated. Row i of $\mathbf{L}$ and A_{ii} are set to 0 before starting. The algorithm keeps track of a list of ancestors $\mathbf{ANC}_i$, whose contribution to A_{ii} has yet to be included. If the sire or dam is unknown $s_i = 0$ or $d_i = 0$, respectively.

$\mathbf{F}_0 = -1$
For $i = 1$ to N (all rows of $\mathbf{L}$)
  $\mathbf{ANC}_i = (i)$; $L_{ii} = 1$; $D_{ii} = [1/2 - 1/4(F_{s_i} + F_{d_i})]$;
  Do while $\mathbf{ANC}_i$ is not empty:
    $j = \max(\mathbf{ANC}_i)$ (j is the youngest animal in $\mathbf{ANC}_i$)
    if $s_j \neq 0$ (sire known): add s_j to $\mathbf{ANC}_i$; and add $1/2 L_{ij}$ to L_{is_j}
    if $d_j \neq 0$ (dam known): add d_j to $\mathbf{ANC}_i$; and add $1/2 L_{ij}$ to L_{id_j}
    add $L_{ij}^2 D_{jj}$ to A_{ii}

delete j from $\mathbf{ANC}_i$
end while
$\mathbf{F}_i = A_{ii} - 1$

The youngest animal j in $\mathbf{ANC}_i$ is evaluated, because all of its progeny in the pedigree of animal i must have been evaluated, hence, its L_{ij} value is known. The kernel of a Fortran program of the algorithm is given in the *Appendix*. The list of ancestors is represented by a link list. In the computer program time is saved by: i) checking whether one of the parents is unknown, giving an inbreeding coefficient of zero (otherwise, the algorithm would trace the pedigree of the other parent); and ii) if the pedigree is sorted such that full sibs have successive identification numbers, *ie* are evaluated successively, only the first full sib is evaluated and all full sibs obtain (and have) the same inbreeding coefficient as the evaluated full sib.

In order to compare the algorithms, Golden *et al*'s (1991) and Tier's (1990) algorithms were programmed in Fortran. Golden *et al* presented their algorithm in **C**, which they considered faster. However, **C** was not available to the authors. The sorting of the list of descendants in the Golden *et al* algorithm was performed by an IMSL routine here (IMSL, 1984).

SIMULATION

In order to compare the algorithms, the simulation program of Meuwissen and Van der Werf (1991) was used to generate a pedigree data set. The program simulated an open dairy cattle nucleus scheme for 10, 20 or 40 years, with population sizes of 9 267, 15 582 and 28 171 animals, respectively (see table I). Selection was for animal model BLUP breeding values. Nucleus dams and commercial dams produced 8 and 1 offspring, respectively. The number of nucleus and commercial sires selected was 4 and 10, respectively. Also, a breeding program was simulated for 20 years with selection of only one commercial sire, representing a situation with large half sib families.

Table I. Parameters of the open nucleus dairy cattle breeding scheme.

No of nucleus animals born per year	128
No of commercial animals born per year	500
Size of paternal half-sib family in nucleus	32
Size of full-sib family in nucleus	8
Size of paternal half-sib family in commercial population	50 or 500
Size of full-sib family in commercial population	1
No of animals after 10, 20 and 40 yr	9 267, 15 582, 28 171

Inbreeding coefficients were calculated on a micro-VAX 3800 in batch mode. The maximum physical memory allocated to the batch queue was about 20 Mb. Because this exceeded the memory required by any of the algorithms, no disk memory was used. If disk memory were used, the computational speed would depend on the size of the physical memory of the computer.

The number of generations in the data-set after 10, 20 and 40 years were counted and a weighted average per individual was calculated, where weighting was according to the contribution of the path. For instance, if only the sire of the sire and the sire of an animal were known, the average number of generations was $2 \cdot 1/4 + 1 \cdot 1/4 + 0 \cdot 1/2 = 0.75$. The animal with the maximum average number of generations and the average number of generations of the animals born in the last year evaluated were estimated.

RESULTS

The results are presented in table II. Although the algorithm of Golden *et al* (1991) and that presented here require the same number of additions of $L_{ij}^2 D_{jj}$ terms, the latter consumed less computer time. This is because the algorithm of Golden *et al*: i) makes a list of offspring of each animal, before starting evaluations; ii) traces descendants instead of ancestors, which is more difficult because the number of progeny of each animal is unlimited, whereas the number of parents is limited to 2; iii) sorts the list of descendants for every animal, which is time-consuming; and iv) tracing of descendants, sorting them, and addition of $L_{ij}^2 D_{jj}$ are not simultaneously performed. The Golden *et al* algorithm requires more memory, because the list of offspring needed and the tracing of descendants requires memory. However, the memory required is still linear with the number of animals N.

Tier's (1990) algorithm was faster than the algorithm presented here, when 40 years are evaluated, *ie* evaluation of 12.3 generations of animals. When many generations are evaluated, many animals have common ancestors, whose pedigree is re-evaluated many times in the present algorithm. Tier's algorithm avoids redundant calculation by tracing pedigrees once. The memory required is 15.9 Mb for 28171 animals. The presented algorithm required 0.7 Mb is this situation.

When about 6 or fewer generations are evaluated, the present algorithm is faster than Tier's, because Tier's algorithm first determines which elements of $\mathbf{A}$ are required before calculating inbreeding coefficients. If only the animals born in year 40 are evaluated, the algorithm presented here and that of Tier (1990) have about the same speed (table II). In Tier's algorithm, the required elements of $\mathbf{A}$ have to be determined for relatively few animals. The time required by the algorithm of Golden *et al* (1991) equals that required if all animals are evaluated, because the algorithm re-calculates all inbreeding coefficients. The last alternative in table II shows that the presence of large half sib families is advantageous to Tier's algorithm. This is because Tier's algorithm traces the pedigree of the sire only once.

DISCUSSION AND CONCLUSIONS

The presented algorithm for computing diagonal elements of $\mathbf{A}$ is a sparse implementation of an algorithm presented by Mrode and Thompson (1989) and Quaas (1989) for the multiplication of $\mathbf{A}$ with a matrix. Here, this matrix is the identity matrix and only diagonal elements are calculated.

The algorithm combines high computational speed with low memory requirement, if the number of generations evaluated is not larger than, say 12 (table II). Hence, the algorithm is suited for large population sizes.

Table II. Computer time and memory requirement of inbreeding algorithms. G, T and M denote Golden *et al*'s (1991); Tier's (1990) and the presented algorithm respectively.

Years evaluated	*Average/ max no of generations*[1]	*Average inbreeding coefficient*[1]	*Algorithm*	*Time (s)*	*Memory (Mb)*
		Size of paternal half sib families is 50			
0 – 10	3.3/3.9	0.055	G	2.7	0.4
			M	0.7	0.2
			T	1.5	1.9
0 – 20	5.8/6.6	0.150	G	12.8	0.7
			M	3.7	0.4
			T	4.1	5.0
0 – 40	12.3/13.2	0.626	G	56.5	1.2
			M	15.5	0.7
			T	8.6	15.9
19 – 20	5.8/6.6	0.150	G	12.8	0.7
			M	0.6	0.4
			T	1.1	5.0
39 – 40	12.3/13.2	0.626	G	56.5	1.2
			M	1.3	0.7
			T	1.8	15.9
		Size of paternal half sib families is 500			
0 – 20	5.9/7.1	0.290	G	14.0	0.7
			M	2.7	0.4
			T	2.8	5.0

[1] Calculated for animals born in the last year evaluated.

In order to keep the calculations within the physical memory of the computer, none of the populations evaluated in table II were really large. If the data set were much larger, Tier's (1990) algorithm especially would require disk memory which would decrease its speed considerably. However, the presence of large half sib families favours the use of Tier's algorithm (table II). If the number of generations involved exceeds, say 12, the algorithm presented here becomes slow compared to Tier's, because common ancestors are traced many times. In order to overcome this problem we may compute $\mathbf{F}_i = 1/2A_{s_i d_i} = \sum_k L_{s_i k} L_{d_i k} D_{kk}$ and store the sire's row of $\mathbf{L}$ ($L_{s_i k}, k = 1, \ldots, s_i$). However, calculating the dam's row of $\mathbf{L}$ costs approximately the same amount of computer time as calculating the individual's row of $\mathbf{L}$, as in [1].

In practice, often inbreeding coefficients of many animals in the population have been calculated. Only those of a few new born animals are unknown. The presented algorithm is suited for these situations, because it does not re-compute inbreeding coefficients.

APPENDIX

Integer arrays (N is the number of animals in the population):

PED(1:2, 1:N) PED(1, i) = sire identification no of animal i
PED(2, i) = dam identification no of animal i

POINT(1:N) POINT (i)= the next oldest ancestor in the link list
= 0 if i is the last ancestor

Real Arrays:

F(0: N) **F**(i) = inbreeding coefficients of animal i. **F**(0) = −1 gives appropriate within family variances for unknown parents by the formula: .5−.25∗(**F** (PED(1, i)+PED(2,i)). **F**(i) is initially set equal to −1, which is more accurate than calculating **F** + 1 and then subtracting 1.

L(1:N) **L**(j) = element ij of matrix **L**, if animal i is evaluated

D(1:N) **D**(i) = within family variance of animal i

Kernel of the Fortran code of the algorithm:

```
F(0)=−1.0                    ! gives within family var for unknown parents
DO I=1,N                              ! for all animals in the population
    IS=PED(1,I)
    ID=PED(2,I)
    PED(1,I)=MAX(IS,ID)                    ! Store parents so that
    PED(2,I)=MIN(IS,ID)                    ! PED(1,i) > PED(2,i)
    D(I)=.5-.25*(F(IS)+F(ID))              ! within family variance
    IF(IS.EQ.0.OR.ID.EQ.0)THEN                  ! non inbred animal
      F(I)=0.0
    ELSE IF(PED(1,I-1).Eq.PED(1,I).and.PED(2,I-1).Eq.PED(2,I))THEN
        F(I)=F(I-1)                  ! i and i-1 are full sibs: F(i)=F(i-1)
    ELSE                                                ! compute F(i)
        FI=−1.0                          ! put F=−1 and add diag of A:
        L(I)=1.0                                  ! initialise row of L
        J=I                              ! j = oldest ancestor of i in list
        DO WHILE (J.NE.0)          ! stop when list of ancestors is empty
            K=J                            ! K is a temporary variable
            R=0.5*L(K)                     ! determine contrib to parents
            KS=PED(1,K)                    ! determine parents of ancestor
            KD=PED(2,K)
            IF(KS.GT.0)THEN                ! find slot in link list for sire
              DO WHILE (POINT(K).GT.KS)    ! stop if next ancestor
                K=POINT (K)                ! is older than sire
              END DO                       ! or is the sire
              L(KS)=L(KS)+R                ! add contribution to sire
              IF(KS.NE.POINT(K))THEN       ! include sire in link list
                POINT(KS)=POINT(K) ! POINT(KS)=next ancest in list
                POINT(K)=KS        ! POINT(previous ancestor)=sire
              END IF
              IF(KD.GT.0)THEN              ! do the same for the dam
```

```
                DO WHILE (POINT(K).GT.KD) ! KS > KD, hence do not
                    K=POINT(K)            ! need to reinitialise
                En do                     ! variable K
                L(KD)=L(KD)+R
                IF(KD.NE.POINT(K))THEN
                    POINT(KD)=POINT(K)
                    POINT(K)=KD
                End if
                    END IF
              END IF
              FI=FI+L(J)*L(J)*D(J)          ! add L*D*L value of animal j
              L(J)=0.0                      ! clear L(j) for next F evaluation
              K=J                           ! store 'old' J in temporary variable K
              J=POINT(J)                    ! 'new' J = next oldest animal in list
              POINT(K)=0      ! clear POINT('old' J) for next F evaluation
            END DO
            F(I)=FI                                              ! store F
        END IF
END DO
```

ACKNOWLEDGMENTS

We are indebted to Thompson and referees for helpful suggestions and comments. The Milk Marketing Board of England and Wales, the Meat and Livestock Commission and the Ministry of Agriculture, Fisheries and Food are gratefully acknowledged for their sponsorship.

REFERENCES

Golden BL, Brinks JS, Bourdon RM (1991) A performance programmed method for computing inbreeding coefficients from large data sets for use in mixed-model analyses. *J Anim Sci*, 69, 3564-3573

Henderson CR (1976) A simple method for computing the inverse of a numerator relationship matrix used in the prediction of breeding values. *Biometrics* 32, 69-83

IMSL (1984) *International Mathematical and Statistical Libraries.* Houston, TX, ed 9.2

Meuwissen THE, van der Werf JHJ (1991) Effects of heterogeneous within-herd variances on the efficiency or dairy cattle breeding schemes. *42th Ann Meet Eur Assoc Anim Prod, Berlin, Germany*

Mrode R, Thompson R (1989) An alternative algorithm for incorporating the relationships between animals in estimating variance components. *J Anim Breed Genet* 106, 89-95

Quaas RL (1976) Computing the diagonal elements and inverse of a large numerator relationship matrix. *Biometrics* 32, 949-953

Quaas RL (1989) Transformed mixed model equations: a recursive algorithm to eliminate A^{-1}. *J Dairy Sci* 72, 1937-1941

Tier B (1990) Computing inbreeding coefficients quickly. *Genet Sel Evol* 22, 419-430