

ORIGINAL ARTICLE

A fast algorithm for computing inbreeding coefficients in large populations

M. Sargolzaei¹, H. Iwaisaki^{1,2} & J.-J. Colleau³

¹ Course of Environmental Management Science, Graduate School of Science and Technology, Niigata University, Niigata, Japan

² Department of Agro-biology, Faculty of Agriculture, Niigata University, Niigata, Japan

³ Station de génétique quantitative et appliquée, Institut national de la recherche agronomique, Jouy-en-Josas Cedex, France

Correspondence

Mehdi Sargolzaei, Laboratory of Animal Breeding and Reproduction, Department of Agro-biology, Faculty of Agriculture, Niigata University, Ikarashi nino-cho 8050, Niigata-Shi 950-2181, Japan.

E-mail: iws@agr.niigata-u.ac.jp

Received: 6 September 2004;

accepted: 27 April 2005

Summary

Inbreeding coefficients of animals are required in many genetic analyses of livestock records. A modification of Colleau's indirect algorithm to compute inbreeding coefficients in large populations is presented. With overlapping generations, the modified algorithm evaluated all progeny of each sire simultaneously in one back and forth exploration of a reduced pedigree. Simulation for a relatively large number of generations, different number of sires, family sizes and mating designs showed that Colleau's algorithm was faster (from 1.2 to 143 times) than two other algorithms under comparison (Tier, modified Meuwissen and Luo), in all situations investigated. Modifying Colleau's algorithm considerably decreased computation time (from 50 to 89%), resulting in a very fast algorithm. The number of sires mostly affected computational efficiency of the modified algorithm, whereas family size and mating design had virtually no effect. In the updating situation, when only animals born in the last year were evaluated, given known inbreeding coefficients for the other, the modified algorithm was also fast compared with the other three algorithms. Memory requirements for the algorithms were also discussed.

Introduction

Genetic analyses of livestock records using Henderson's mixed model methodology require the inverse of the numerator relationship matrix ($\mathbf{A}^{-1}$), in which individual inbreeding coefficients are necessary. The individual inbreeding coefficients are also required in order to determine the actual mean level of inbreeding to optimize selection and matings for long-term genetic gain and to manage genetic diversity. Until now, a variety of algorithms for computing inbreeding coefficients were developed (e.g. Wright 1922; Emik & Terrill 1949; Henderson 1976; Quaas 1976; Tier 1990; Golden *et al.* 1991; Meuwissen & Luo 1992; Colleau 2002). However, with large

populations, corresponding memory or time requirements might be important factors to consider. Among these algorithms, Tier's (1990), Meuwissen & Luo's (1992) and Colleau's (2002) algorithms were found to be computationally more efficient in large populations.

Tier's (1990) algorithm was an efficient implementation of the tabular method presented by Emik & Terrill (1949). It utilized a very small subset of matrix $\mathbf{A}$ to compute its diagonal and was relatively fast because of tracing the pedigree only twice and using an efficient sparse method to store the subset. For very large populations, however, large amount of memory is usually needed. Meuwissen & Luo's (1992) algorithm was based on Henderson's (1976)

decomposition of matrix $\mathbf{A}$ ($\mathbf{TBT}'$). The algorithm traced only non-zero elements of matrix $\mathbf{T}$ row by row constructing a list of ancestors for each animal. The efficiency of this algorithm depended highly on the number of generations involved and on the size of full-sib families. Quaas (1995) modified this algorithm (cited by Mrode 1996). Sargolzaei & Iwaisaki (2004) also presented a modification for this algorithm in which the lists of ancestors for each of animals were obtained by a relatively efficient method using the longest ancestral path numbers. With Colleau's (2002) algorithm, inbreeding coefficients were computed at the same time for the animals of the same combination of sire $\times$ pseudo-generation numbers, exploiting sparseness of matrix $\mathbf{A}^{-1}$ (indirect method). In the case of overlapping generations, progeny of a given sire may have different pseudo-generation numbers, therefore each sire may have to be processed more than once. This algorithm was found to be faster than that of Meuwissen & Luo (1992). The memory requirements of Meuwissen & Luo's (1992) and Colleau's (2002) algorithms are linear with the number of animals and consequently, small even in large populations.

The objectives of the current study were to present a modification of Colleau's (2002) algorithm and to compare its computational properties to those of the original algorithms of Colleau (2002) and Tier (1990) and of the modified algorithm of Meuwissen and Luo (Sargolzaei & Iwaisaki 2004).

Methods

Colleau's (2002) indirect method

Henderson (1976) showed that the inverse of the numerator of relationship matrix could be calculated, as follows:

$$\mathbf{A}^{-1} = (\mathbf{T}^{-1})' \mathbf{B}^{-1} \mathbf{T}^{-1}$$

where $\mathbf{T}^{-1}$ is a lower triangular matrix whose elements are zero except for 1 on the diagonal and -0.5 for elements where parents and their progeny are linked, and $\mathbf{B}^{-1}$ is a diagonal matrix whose diagonal elements are the inverse of the within family segregation variances.

Using this formula, Colleau (2002) developed an indirect method for multiplying any vector by the numerator relationship matrix corresponding to planned matings. His main objective was to calculate relationship statistics for planned matings, where inbreeding coefficients of ancestors were assumed to be available. Here, for simplifying the presentation, we consider a group of animals of interest and their

ancestors, disregarding whether this group can be progeny of planned matings or any set of animals in the pedigree. Animals are assumed to have been sequentially numbered from 1 to n , so that parents precede their progeny. Then $\mathbf{A}^*$ stands for the numerator relationship matrix involving animals of interest and their ancestors. Summation of relationships between each of these animals and animals of interest could be obtained as

$$\mathbf{A}^* \mathbf{x} = \mathbf{y}$$

where vector $\mathbf{x}$ contains 1 for the group considered and 0 otherwise, and the term y_i of vector $\mathbf{y}$ is the sum of relationships between animal i and the group considered. In order to obtain $\mathbf{y}$ without computing matrix $\mathbf{A}^*$, Henderson's (1976) formula was employed, as follows:

$$(\mathbf{T}^{*-1})' \mathbf{B}^{*-1} \mathbf{T}^{*-1} \mathbf{y} = \mathbf{x}.$$

Then, this system of equations was solved for vector $\mathbf{y}$ in two following steps, using the special characteristics of matrix $\mathbf{T}^{*-1}$:

(i) solve the following system for $\mathbf{z}$ by tracing back the pedigree

$$(\mathbf{T}^{*-1})' \mathbf{z} = \mathbf{x}$$

where

$$\mathbf{z} = \mathbf{B}^{*-1} \mathbf{T}^{*-1} \mathbf{y};$$

(ii) solve the following system for $\mathbf{y}$ by tracing forth the pedigree

$$\mathbf{T}^{*-1} \mathbf{y} = \mathbf{B}^* \mathbf{z}.$$

Inbreeding coefficients of the animals were assumed to be known when constructing matrix $\mathbf{B}$.

Finally, for computing the individual inbreeding coefficients themselves, this author proposed the following implementation of this indirect method:

(i) calculate pseudo-generation numbers (ψ), where ψ for progeny = $1 + [\max(\psi \text{ for parents})]$;
 (ii) sort animals based on these numbers in ascending order and recode them sequentially starting from 1;
 (iii) for groups in ascending order of ψ , implement as follows:

- (a) extract the ancestors of the current group and recode them;
- (b) apply the indirect method for each sire represented in the current group ($\mathbf{x}$ contains only 1 for the sire considered);
- (c) obtain F for progeny of the sire as half of the numerator relationships between the sire and his mates.

With the indirect method, the inbreeding coefficients of ancestors were assumed to be available. Therefore, the pseudo-generation numbers were used, so that inbreeding coefficients of ancestors be calculated before those of their progeny (in order to circumvent the problem of overlapping generations). Since with overlapping generations, each sire may have progeny with different pseudo-generation numbers, each sire might have to be processed several times.

The modified algorithm

Let $\mathbf{A}_1$ be the numerator relationship matrix involving the sire considered and his ancestors and $\mathbf{A}_2$ be the numerator relationship matrix involving the sire considered, his mates and their ancestors. Then we have

$$\mathbf{A}_1 \mathbf{x}_1 = \mathbf{y}_1$$

$$\mathbf{A}_2 \mathbf{x}_2 = \mathbf{y}_2$$

where vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ contain 1 for the sire considered and 0 otherwise, and vectors $\mathbf{y}_1$ and $\mathbf{y}_2$ contain the relationships between the sire and other animals considered in $\mathbf{A}_1$ and $\mathbf{A}_2$, respectively. Then

$$(\mathbf{T}_1^{-1})' \mathbf{z}_1 = \mathbf{x}_1$$

$$(\mathbf{T}_2^{-1})' \mathbf{z}_2 = \mathbf{x}_2$$

where $\mathbf{z}_1 = \mathbf{B}_1^{-1} \mathbf{T}_1^{-1} \mathbf{y}_1$ and $\mathbf{z}_2 = \mathbf{B}_2^{-1} \mathbf{T}_2^{-1} \mathbf{y}_2$. It can be shown that after solving these linear systems for $\mathbf{z}_1$ and $\mathbf{z}_2$ by tracing back the pedigree, these two vectors would have the same values for the sire and his ancestors and zero for the other animals in vector $\mathbf{z}_2$. This is because vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ contain 1 only for the sire and zero otherwise. Hence, the terms of vectors $\mathbf{B}_1 \mathbf{z}_1$ and $\mathbf{B}_2 \mathbf{z}_2$ pertaining to the sire and his ancestors are the same and zero otherwise in vector $\mathbf{B}_2 \mathbf{z}_2$. Thus, $\mathbf{B}_2 \mathbf{z}_2$ can be derived from $\mathbf{B}_1 \mathbf{z}_1$ and consequently, all progeny of each sire can be evaluated at one back and forth exploration of the pedigree only knowing inbreeding coefficients for the ancestors of the sire.

Basically, calculating relationship coefficients between a sire and his mates does not require inbreeding coefficients of those mates, which are not ancestors of the sire. Then, mates can be much younger than sire, without affecting calculations. This fact, not exploited by Colleau (2002), can be interpreted in terms of drift. Consider the sire and one mate. The lists of respective ancestors are (anc_1

anc_2) and (anc_1 anc_3) where anc_1 corresponds to common ancestors and anc_2 and anc_3 are consequently not common ancestors. Finally, relationship between the sire and the mate only occur through anc_1 . Then, only the drift corresponding to inbreeding coefficients of anc_1 has to be accounted for.

Finally, the algorithm presented by Colleau (2002) can be modified, as follows:

- (i) extract the pedigree of all sires and dams and recode them sequentially starting from 1;
- (ii) sort identification of animals into an array based on the identification of their sire in ascending order;
- (iii) process the indirect method, sire after sire, as follows:

- (a) compute $\mathbf{B}_1 \mathbf{z}_1$ by tracing back the reduced pedigree;
- (b) compute $\mathbf{y}_2$ by tracing forth the reduced pedigree;
- (c) obtain F for progeny of the sire considered as half of the relationship between the sire and his mates.

With this algorithm, pseudo-generation numbers are not used, therefore there is no need to recode the main pedigree: ancestors of animals are extracted and recoded only once and all progeny of each sire are evaluated simultaneously irrespective of whether generations are discrete or overlapping. The application of the algorithm in the updating situation, where only newly born animals are to be evaluated, is straightforward by extracting the ancestors of the newly born animals and applying the indirect algorithm to their sires. The kernel of a C program of the current algorithm is given in the Appendix.

Simulation study

Stochastic simulations were carried out in order to compare the current algorithm to those of Colleau (2002) and Tier (1990) and one of the modifications of Meuwissen and Luo (Sargolzaei & Iwaisaki 2004) in terms of computation time and memory requirement. The base populations consisted of 100 or 400 sires and 2000 dams that were assumed to be unrelated, non-inbred and unselected. A hierarchical mating design in which each dam was mated to 1 sire and a factorial mating design in which each dam was mated to two or eight sires were considered. In order to keep the family size constant in both mating designs, each dam produced two or eight (hierarchical mating) and one (factorial mating) progeny per mating. Each year, the oldest 50 and 30% of sires and dams, respectively, were replaced with selected progeny from the previous year. Selection was based

on phenotypic values of a trait of initial h^2 equal to 0.4 and selected parents were mated at random. All these schemes were simulated for 20, 40 or 60 generations.

The average number of generations of the animals born in the last year was computed, as described by Meuwissen & Luo (1992). The algorithms compared were programmed using C language. The required memory was assessed taking account of 4, 4 and 2 bytes for *floating-point*, *integer* and *short* variables, respectively. Computations were carried out on a PC Pentium 4, 3.2 GHz processor under Windows 2000 (1 Gb RAM).

Results and discussion

Tables 1 and 2 present the computing properties of the four algorithms compared. Clearly, the algorithm of Colleau (2002) (C) and its modification (MC) were faster than those of Tier (1990) (T) and modification of Meuwissen and Luo (Sargolzaei & Iwaisaki 2004) (MM). Mating design had almost no significant effect on computation time of C and MC. Mating design, however, affected the efficiency of MM significantly. This was because with MM, time was saved when full-sib families existed and there were less full-sib families with factorial rather than hierarchical mating design, as pointed out by Colleau (2002). Method T was less sensitive to the mating design than MM, but had higher efficiency with

hierarchical mating design. Before considering the effect of family size on the efficiency of methods, it should be reminded that in our context, increasing family size corresponded not only to increased overall number of animals across generations (likely to increase computation time) but also to increased selection pressure, hence to decreased number of genetic origin (likely to decrease computation time). The balance between these contradictory effects depended indeed on methods considered. Large family size was advantageous for T. This was also reported by Meuwissen & Luo (1992). Selection led to a non-random distribution of family size in selected animals by selecting related animals. Therefore, the presence of large family size in population under selection was likely to decrease the small subset of **A** for diagonal processing, as compared with small family size. With method MM, the effect of an increased overall number of animals whose pedigree had to be traced back was dominant, because computation time increased when family size increased. This increase was even dramatic in the factorial mating design, because the advantage of considering full-sibs almost disappeared and because very similar pedigrees should have to be traced back. With hierarchical mating design and discrete generations, however, family size might not affect the efficiency of MM so much. Generally, MM became very slow when the number of generations increased or when full-sib family size decreased. This was because with MM,

Table 1 Computing properties of the algorithms for populations with 100 base sires

Mating design ^a	Family size	Years evaluated	Number of animals	Average number of generations ^b	Computing time (s)				Memory requirement (Mb)			
					T	MM	C	MC	T	MM	C	MC
H	2	0–20	82 100	12.7	3.0	1.2	0.2	0.1	27.9	2.0	2.8	1.8
		0–40	162 100	25.3	8.0	29.3	1.3	0.3	68.9	4.0	5.5	3.6
		0–60	242 100	37.9	13.2	95.0	3.5	0.8	110.1	6.0	8.2	5.4
		59–60	4000	37.9	8.6	4.3	0.6	0.1	80.9	6.0	8.2	5.4
	8	0–20	322 100	14.3	2.0	2.5	0.6	0.1	27.1	8.0	10.6	6.4
		0–40	642 100	28.5	5.1	41.5	2.8	0.4	61.6	15.9	21.0	12.8
		0–60	962 100	42.7	8.3	127.8	7.1	0.8	95.9	23.9	31.5	19.1
		59–60	16 000	42.7	4.2	5.6	1.7	0.1	69.3	23.9	31.5	19.1
	F	0–20	82 100	12.7	3.6	2.3	0.2	0.1	32.6	2.0	2.8	1.8
		0–40	162 100	25.2	9.8	61.0	1.3	0.3	79.8	4.0	5.5	3.6
		0–60	242 100	37.7	16.0	197.5	3.7	0.8	127.6	6.0	8.2	5.4
		59–60	4000	37.7	9.9	8.8	0.6	0.1	88.8	6.0	8.2	5.4
	8	0–20	322 100	14.2	4.8	23.1	0.7	0.1	45.8	8.0	10.6	6.4
		0–40	642 100	28.3	10.9	366.3	3.1	0.5	98.3	15.9	21.0	12.8
		0–60	962 100	42.4	18.3	1115.2	7.8	1.0	151.5	23.9	31.5	19.1
		59–60	16 000	42.4	6.2	47.9	1.8	0.2	83.1	23.9	31.5	19.1

T, Tier's algorithm; MM, modification of Meuwissen and Luo's algorithm; C, Colleau's algorithm; MC, modification of C.

^aH, hierarchical; F, factorial.

^bCalculated for animals born in the last year evaluated.

Table 2 Computing properties of the algorithms for populations with 400 base sires

Mating design ^a	Family size	Years evaluated	Number of animals	Average number of generations ^b	Computing time (s)				Memory requirement (Mb)			
					T	MM	C	MC	T	MM	C	MC
<i>H</i>	2	0–20	82 400	12.3	16.1	2.3	0.8	0.3	86.9	2.0	2.8	1.9
		0–40	162 400	24.5	43.6	48.4	4.4	1.6	221.8	4.0	5.5	3.7
		0–60	242 400	36.7	72.1	151.6	11.5	4.0	358.0	6.0	8.2	5.6
		59–60	4000	36.7	65.5	6.7	1.2	0.2	332.6	6.0	8.2	5.6
	8	0–20	322 400	13.5	9.0	5.2	2.2	0.4	62.4	8.0	10.6	6.5
		0–40	642 400	27.1	24.3	74.9	8.8	1.6	152.9	15.9	21.1	12.9
		0–60	962 400	40.6	40.0	233.6	20.4	4.1	245.1	23.9	31.6	19.3
		59–60	16 000	40.6	35.1	10.3	2.5	0.3	213.9	23.9	31.6	19.3
<i>F</i>	2	0–20	82 400	12.3	18.8	4.4	1.0	0.3	98.0	2.0	2.8	1.9
		0–40	162 400	24.4	49.6	97.8	5.5	1.6	242.9	4.0	5.5	3.7
		0–60	242 400	36.4	80.8	306.4	13.7	4.1	387.5	6.0	8.2	5.6
		59–60	4000	36.4	74.8	13.6	1.3	0.2	353.4	6.0	8.2	5.6
	8	0–20	322 400	13.5	17.9	46.5	3.0	0.4	101.4	8.0	10.6	6.5
		0–40	642 400	27.1	41.3	677.5	11.2	1.8	225.9	15.9	21.1	12.9
		0–60	962 400	40.7	67.7	2079.3	25.7	4.4	349.3	23.9	31.6	19.3
		59–60	16 000	40.7	44.0	89.7	2.8	0.3	261.0	23.9	31.6	19.3

T, Tier's algorithm; MM, modification of Meuwissen and Luo's algorithm; C, Colleau's algorithm; MC, modification of C.

^a*H*, hierarchical; *F*, factorial.

^bCalculated for animals born in the last year evaluated.

the pedigree of each animal, except for full-sibs, was traced back separately, and the average number of ancestors per animal in population with the large number of generations was usually large. Method C became slow as family size increased, because Colleau's extraction of pedigree for each pseudo-generation in large populations required more time than it did in small populations. It was found that MC had almost the same speed irrespective of family size, because the pedigree of all sires and dams were extracted once and because full-sibs and half-sibs from each sire were evaluated simultaneously. All algorithms became slow, when the number of sires increased and, indeed, the number of sires turned out to be the most important factor influencing computation time of T, C and MC. For example, computation time for 400 sires was 3.7–8.4, 1.5–5.0 and 1.5–5.3 times as large as for 100 sires, for T, C and MC, respectively. The computation time of the current modified algorithm was 50–89% lower compared with that of C. Reasons of this improved efficiency were as follows: (i) extracting and recoding pedigree of all ancestors (reduced pedigree) were carried out only once; and (ii) the back and forth exploration of the reduced pedigree only occurred once for each sire even in the situation of overlapping generations (as for C in separate generations).

With this simulation, the memory requirements of MC and T were the smallest and the largest, respectively, and MM, C and MC required substantially less

memory compared with T. With T, memory requirement similar to computation time depended on the small subset of matrix **A** and it exhibited almost the same trend as for computation time. It should be noted that memory requirement for MM is a linear function of only the total number of animals.

All of methods, C, MC, T and MM, did not recompute known inbreeding coefficients. In the case when inbreeding coefficients were unknown only for newly born animals in year 20, 40 or 60, MC was also found to be faster than the other three algorithms (only results for year 60 shown in Tables 1 and 2). In this case, the rate of decrease in computation times of MM, C and MC increased relative to the evaluation of animals over all years, as the number of generations and years evaluated increased. The efficiency of MM highly depended on the proportion of new animals. In contrast, the efficiency of C and MC depended instead on the proportion of the sires of these animals in the overall sire population. The efficiency of T also increased, especially in the case of 100 sires, but not so substantially as for the other algorithms compared. However, the rate of decrease in memory requirement and computation time of T for year 20, 40 and 60 within each factor considered were not significantly different. For the evaluation of the newly born animals, MM, C and MC required the same amount of memory as in the evaluation of animals over all years. Although the memory requirement of

T was lower than that for evaluation of animals over all years, it turned out to be still very high compared with MM, C and MC.

Conclusion

Inbreeding coefficients of animals are required in many genetic analyses of livestock records. Colleau's algorithm, which is derived from his indirect method for calculating relationship coefficients, was modified in order to compute inbreeding coefficients faster in large populations with overlapping generations. We showed that with the indirect method in situation of overlapping generations, inbreeding coefficients for each paternal group can be calculated at one back and forth exploration of the pedigree only based on inbreeding coefficients of sire's ancestors. Extracting and recoding of pedigree of ancestors for each paternal group, separately, are time-consuming tasks: with the currently modified algorithm, a reduced pedigree, which contains the ancestors of all animals, is extracted and recoded only once.

Acknowledgement

The authors thank the Wagyu Cattle Registry Association for partial financial support.

References

- Colleau J.J. (2002) An indirect approach to the extensive calculation of relationship coefficients. *Genet. Sel. Evol.*, **34**, 409–421.
- Emik L.O., Terrill, C.E. (1949) Systematic procedures for calculating inbreeding coefficients. *J. Heredity*, **40**, 51–55.
- Golden B.L., Brinks J.S., Bourdon R.M. (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 C.R. (1976) A simple method for computing the inverse of a numerator relationship matrix used in the prediction of breeding values. *Biometrics*, **32**, 69–83.
- Meuwissen T.H.E., Luo Z. (1992) Computing inbreeding coefficients in large populations. *Genet. Sel. Evol.*, **24**, 305–313.
- Mrode R.A. (1996) *Linear Models for the Prediction of Animal Breeding Values*. CAB International, Wallingford, UK.
- Quaas R.L. (1976) Computing the diagonal elements and inverse of a large numerator relationship matrix. *Biometrics*, **32**, 949–953.
- Quaas R.L. (1995) *Fx Algorithms*. An Unpublished Note, Cornell University, Ithaca, NY.
- Sargolzaei M., Iwaisaki H. (2004) An efficient algorithm for computing inbreeding coefficients in large populations. *Jpn. J. Biometrics*, **25**, 25–36.
- Tier B. (1990) Computing inbreeding coefficients quickly. *Genet. Sel. Evol.*, **22**, 419–430.
- Wright S. (1922) Coefficients of inbreeding and relationships. *Am. Nat.*, **56**, 330–338.

Appendix

C code for the modified algorithm

n = number of animals in total; m = number of sires and dams in total

Variables declaration

```
int i, j, k, rN, rS, S, D, MIP; // integer variables.
int Ped[n + 1][2], rPed[m + 1][2]; // main and reduced pedigrees, respectively
int SId[n + 1]; // will contain the sorted animals ID based on the ID of their sires
int Link[n + 1]; // will contain new ID of ancestors at position of their original ID
int MaxIdP[m + 1]; // will contain maximum new ID of parents for each paternal
// group at position of the new ID of each sire
float F[n + 1], B[m + 1], x[m + 1]; // inbreeding coefficients, within family
// segregation variances and x arrays, respectively
```

Kernel of the modified algorithm

```
F[0] = -1.0f, x[0] = 0.0f, Link[0] = 0; // set values for the unknown parent
for(rN = i = 1; i <= n; ++i){ // extract and recode ancestors
    SId[i] = i, Link[i] = 0; if(i <= m) x[i] = 0.0f; // initialization
    S = Ped[i][0], D = Ped[i][1];
    if(S && !Link[S]){
        MaxIdP[rN] = Link[S] = rN;
        rPed[rN][0] = Link[Ped[S][0]];
        rPed[rN++][1] = Link[Ped[S][1]];
    }
    if(D && !Link[D]){
        Link[D] = rN;
        rPed[rN][0] = Link[Ped[D][0]];
        rPed[rN++][1] = Link[Ped[D][1]];
    }
    if(MaxIdP[Link[S]] < Link[D]) MaxIdP[Link[S]] = Link[D]; // determine
    // maximum ID of parents for each paternal group
}
```

```
MyQuickSort(Ped, SId); // sort animals according to ID of their sires into SId
for(k = i = 1; i <= n; ++i){ // do from the oldest sire to the youngest sire
    if(!Ped[SId[i]][0]) F[SId[i]] = 0.0f; // sire is not known
    else{
        S = Ped[SId[i]][0], rS = Link[S], MIP = MaxIdP[rS];
        x[rS] = 1.0f;
        for(k = S; k <= S; ++k) // compute within family segregation variances
            if(Link[k]) B[Link[k]] = 0.5f - 0.25f * (F[Ped[k][0]] + F[Ped[k][1]]);
        for(j = rS; j > -j){ // trace back the reduced pedigree
            if(x[j]){ // consider only ancestors of the sire
                if(rPed[j][0]) x[rPed[j][0]] += x[j] * 0.5f;
                if(rPed[j][1]) x[rPed[j][1]] += x[j] * 0.5f;
                x[j] * = B[j];
            }
        }
        for(j = 1; j <= MIP; ++j) // trace forth the reduced pedigree
            x[j] += (x[rPed[j][0]] + x[rPed[j][1]]) * 0.5f;
        for(i = n; i <= n; ++i) // obtain F for progeny of the current sire
            if(S != Ped[SId[i]][0]) break;
        else F[SId[i]] = x[Link[Ped[SId[i]][1]]] * 0.5f;
        for(j = 1; j <= MIP; ++j) x[j] = 0.0f; // set to 0 for next evaluation of sire
    }
}
```