Journal of
Animal Breeding and Genetics

J. Anim. Breed. Genet. ISSN 0931-2668

ORIGINAL ARTICLE

# Efficient computation of the genomic relationship matrix and other matrices used in single-step evaluation

I. Aguilar[1,2], I. Misztal[1], A. Legarra[3] & S. Tsuruta[1]

1 Animal and Dairy Science Department, University of Georgia, Athens, GA, USA
2 Instituto Nacional de Investigación Agropecuaria, INIA Las Brujas, Uruguay
3 INRA, UR631 SAGA, BP 52627, 32326 Castanet-Tolosan, France

**Keywords**
Computing methods; genomic selection; relationship matrix.

**Correspondence**
I. Aguilar, Instituto Nacional de Investigación Agropecuaria, INIA Las Brujas, Ruta 48 km 10, Canelones, CP 90200, Uruguay. Tel: +598 2 3677641; Fax: +598 2 367 7609;
E-mail: iaguilar@inia.org.uy

Received: 2 June 2010;
accepted: 5 October 2010

## Summary

Genomic evaluations can be calculated using a unified procedure that combines phenotypic, pedigree and genomic information. Implementation of such a procedure requires the inverse of the relationship matrix based on pedigree and genomic relationships. The objective of this study was to investigate efficient computing options to create relationship matrices based on genomic markers and pedigree information as well as their inverses. SNP maker information was simulated for a panel of 40 K SNPs, with the number of genotyped animals up to 30 000. Matrix multiplication in the computation of the genomic relationship was by a simple 'do' loop, by two optimized versions of the loop, and by a specific matrix multiplication subroutine. Inversion was by a generalized inverse algorithm and by a LAPACK subroutine. With the most efficient choices and parallel processing, creation of matrices for 30 000 animals would take a few hours. Matrices required to implement a unified approach can be computed efficiently. Optimizations can be either by modifications of existing code or by the use of efficient automatic optimizations provided by open source or third-party libraries.

## Introduction

Genomic evaluations in dairy cattle are currently performed using multiple-step procedures (Hayes *et al.* 2009; VanRaden *et al.* 2009). A typical genomic evaluation requires a traditional evaluation with an animal model, extraction of pseudo-observations such as deregressed evaluations or daughter deviations, estimation of genomic effects for genotyped animals, and their combination with traditional parent averages and breeding values (Hayes *et al.* 2009; VanRaden *et al.* 2009). Genomic effects can be estimated with a simple model that includes a genomic relationship matrix derived from genotypes and variances of the SNP marker effects (Nejati-Javaremi *et al.* 1997; VanRaden 2007). It is worth noting that these genomic relationship matrices are also used for ancestry correction in genome-wide association studies (Astle & Balding 2009; Kang *et al.* 2010).

Recently, Misztal *et al.* (2009) proposed a single-step evaluation in which the pedigree-based relationship matrix is augmented by contributions from the genomic relationship matrix. Legarra *et al.* (2009) derived a joint relationship matrix based on pedigree and genomic relationships, and Aguilar *et al.* (2010) described the inverse of the joint relationship matrix. A similar matrix and its inverse were independently derived by Christensen & Lund (2010).

VanRaden (2008) discussed methods to create genomic relationship matrices. The kernel of such methods involves multiplication of the matrix of marker incidences (with dimension equal to number of genotyped animals by number of SNP markers) by its transpose. Matrix operations can be implemented efficiently using linear algebra kernels called Basic Linear Algebra Subroutines (BLAS; http://www.netlib.org/blas) (Dongarra *et al.* 1988, 1990b). Optimized BLAS subroutines were developed by Whaley & Dongarra (1998), and an open source of these libraries is available as Automatically Tuned Linear Algebra Software (ATLAS; http://math-atlas.sourceforge.net). These libraries account for properties of a specific processor (memory speed and cache size).

Modifications of current software to implement the single-step approach (Aguilar *et al.* 2010) require inverses of the genomic relationship matrix (**G**) and the pedigree-based numerator relationship matrix ($\mathbf{A}_{22}$) for genotyped animals. The objective of this research was to present efficient computing options to create these matrices and their inverses.

## Materials and methods

The inverse of the relationship matrix ($\mathbf{H}^{-1}$) based on both pedigree and genomic information (Aguilar *et al.* 2010) is

$$\mathbf{H}^{-1} = \mathbf{A}^{-1} + \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{G}^{-1} - \mathbf{A}_{22}^{-1} \end{bmatrix}$$

where $\mathbf{A}^{-1}$ is the inverse of the numerator relationship matrix for all animals, $\mathbf{G}^{-1}$ is the inverse of the genomic relationship matrix for genotyped animals, and $\mathbf{A}_{22}^{-1}$ is the inverse of the numerator relationship matrix for genotyped animals. Modifications of current software for genetic evaluation (Misztal *et al.* 2002; Tsuruta *et al.* 2001) require the computation of $\mathbf{G}^{-1}$ and $\mathbf{A}_{22}^{-1}$ to use $\mathbf{H}^{-1}$ instead of $\mathbf{A}^{-1}$.

For the purpose of testing the computation of the genomic matrix, an incidence matrix of 40 K SNP marker information (**M**) was simulated, with values corresponding to gene content of the second allele (0, 1 and 2) sampled from a uniform distribution. The number of simulated genotyped animals varied from 1000 to 30 000, and no relationship was assumed. Matrix $\mathbf{A_{22}}$ was constructed using the pedigree of 9 100 106 US Holsteins provided by Holstein USA Inc. (Brattleboro, VT, USA).

Following VanRaden (2008), a centred matrix (**Z**) was constructed by subtracting from M the matrix P = 2p′1 containing the expected genotype frequencies corresponding to allele frequencies p; then **G** was constructed as

$$\mathbf{G} = \frac{\mathbf{ZZ}'}{k}.$$

where the scaling parameter $k$ was

$$k = 2\sum p_j(1 - p_j).$$

### Computer memory hierarchy

In current computers, main memory is accessed via a much faster but also much smaller cache memory (Dongarra *et al.* 1990a). In operations limited by computer memory access, speed is optimized by maximizing reuse of data in cache memory and minimizing traffic from the main memory. As various computers differ by size and characteristics of cache and main memory, optimizations are computer dependent. The optimal use of the memory hierarchy is important especially for computers with multiple processors, where each processor has its own cache, but the main memory is shared between processors (Whaley & Dongarra 1998).

### Methods

Computations of $\mathbf{ZZ}'/k$ were coded in Fortran 95 by 3 methods (Figure 1). The first method (ORIG) consisted of nested 'do' loops, where centring the matrix **Z** was through indirect memory access, and matrix **Z** was not set up explicitly. In the second method (OPTM), the matrix **Z** was centred outside the main loop to avoid indirect addressing. In the third method (OPTML), the main loop was split into two loops. In each method, a table with appropriate coefficients for centring, **W**, was precomputed. This table contains three rows indexed by 0, 1 and 2 (i.e. gene content) and the j-th column contains $-2p_j$, $1-2p_j$, $2-2p_j$, that is, the coefficients that are used in **Z** without computing **P** explicitly. Having separate operations for matrix multiplication and scaling allows the use of standard subroutines to compute $\mathbf{ZZ}'$. Matrix multiplications of the form $\mathbf{ZZ}'$ were computed by the original BLAS subroutine DGEMM and by its optimized versions as in ATLAS or in the Intel Math Kernel Library (MKL).

The scaling parameter defined as $k = 2\sum p_j(1 - p_j)$ assume that there is no missing genotype. Formulae to account for missing genotypes per animal were presented by VanRaden (2008). Codes presented in the Figure 1 implemented such formulae using a vector **k**, with dimension of number of genotyped animals, with elements $\mathbf{k}_i = 2\,\Sigma$

**M** = *n* by *m* incidence matrix of SNP markers, tM = M', *n* = number of animals, *m* = number of markers; W = centered matrix with dimension 3 by *m* with values corresponding to gene content - $2p_l$ ; $p_l$ = allele frequencies at locus *l*; ***k*** = n by 1 vector of scaling parameters.

**Original (ORIG)**

```
do i=1,n
   do j=i,n
      S=0
      do l=1,m
          S=S+W(tM(l,i),l)
             *W(tM(l,j),l)
      end do
      G(i,j)= S /
             sqrt(k(i)*k(j))
      G(j,i)=G(i,j)
    end do
end do
```

**Optimize indirect memory access (OPTM)**

```
do l=1,m
   tZ(l,:)=W(tM(l,:),l)
end do
do i=1,n
   do j=i,n
      S=0
      do l=1,m
         S=S+tZ(l,i)
            *tZ(l,j)
      end do
      G(i,j)= S /
             sqrt(k(i)*k(j))
      G(j,i)=G(i,j)
   end do
end do
```

**Optimize memory & loops (OPTML)**

```
G=0
do l=1,m
   Z(:,l)=W(M(:,l),l)
end do
do i=1,n
   do j=1,n
      do l=1,m
         G(i,j)=G(i,j)
               +Z(i,l)*Z(j,l)
      end do
   end do
end do
do i=1,n
   do j=1,n
      G(i,j)= G(i,j) /
             sqrt(k(i)*k(j))
   end do
end do
```

**Figure 1** Alternative codes to compute ZZ'/k.

$p_j(1 - p_j)$ with the summation over the non-missing loci of animal *i*.

The $\mathbf{A}_{22}$ should include information on all ancestors of genotyped animals. This matrix was computed by two algorithms. The first one by the tabular method (Emik & Terrill 1949) requires a matrix of the size of all involved animals. The second one by Colleau (2002) implemented as in Misztal *et al.* (2009) requires only a matrix for genotyped animals. The source code of the algorithm in Fortran 95 is shown in the Appendix.

Inversions of matrices **G** and $\mathbf{A}_{22}$ were computed by converted Fortran 95 code of the generalized inverse algorithm from the BLUPF90 package (Misztal *et al.* 2002), which is based on the Cholesky decomposition, and by the LU factorization using the DGETRF/DGETRI subroutines from LAPACK (Anderson *et al.* 1990). The Cholesky decomposition could be also performed using the subroutines DPOTRF/DGETRI from LAPACK; both subroutines are available either in ATLAS or in MKL libraries. If **G** is singular, e.g. with clones, VanRaden (2008) proposed using a weighted matrix: $\mathbf{G}w = w*\mathbf{G} + (1 - w)*\mathbf{A}_{22}$.; weights of 0.95 and 0.98 yielded nearly identical EBV in Aguilar *et al.* (2010).

## Computations

All programs were run on an Opteron 64-bit processor with a clock speed of 3.02 GHz and a cache size of 1 Mbyte. Some programs for **G** matrix inversion were run on a Xeon 64-bit processor with a clock speed of 3.5 GHz and a cache size of 6 Mbyte. Initial software for the construction of **G** and for the tabular method was provided by P.M. VanRaden (USDA-ARS, Animal Improvement Programs Laboratory, Beltsville, MD,USA). Fortran programs were compiled by Intel Fortran Compiler with option -O3. For parallel processing, automatic parallelization was achieved by compiling with options that selected the OpenMP package (http://www.openmp.org) implemented in the MKL libraries.

## Results and discussion

Computing times for alternative codes to create the **G** matrix using two types of processing systems are presented in Table 1. The ORIG method was 10 times slower on the Opteron system than on the Xenon system. Large improvement was achieved with OPTM on the Opteron but not on the Xeon system. Faster computing on the Xeon system could be because of larger cache size and different processor characteristics. Also, the Intel compiler could work more efficiently in optimizing codes on Xeon than on Opteron systems. Approximately four times speedup was obtained on both computers with OPT-ML, which is the code optimized for memory and with loop rearrangement. Interestingly, in OPTML, the whole construction of **G** is split into separate, smaller problems: first build **Z**, compute **ZZ**' and standardize it. Clearly, this facilitates an automatic optimization. Theses alternative codes were tested using other compilers (Absoft and GNU Fortran) in the Opteron system (results not shown). Similar pattern in speedup was obtained with the Absoft

**Table 1** Computing time (m) for alternative codes for the creation of the genomic relationship matrix using two types of processing systems[1]

| | | Algorithms | | |
|---|---|---|---|---|
| Processor (clock speed) | Cache memory size | ORIG | OPTM | OPTML |
| Xeon (3.5 GHz) | 6 Mbyte | 24 | 26 | 7 |
| Opteron (3.02 GHz) | 1 Mbyte | 265 | 59 | 17 |

ORIG, Original code for creation the genomic relationship matrix; OPTM, Optimized code to avoid indirect memory addressing; OPTML, Optimized code to avoid indirect memory addressing and loop reordering.

[1]Using 6500 animals and 40 K SNP markers.

compiler. With the GNU Fortran, the vectorization was not obtained and OPTML performed poorly.

Different algorithms to perform the matrix multiplication were tested with the Opteron system. Figure 2 shows the computing time for the BLAS subroutine DGEMM, the modified code for memory access and loop ordering (OPTML), and the DGEMM optimized version as in ATLAS library. Relative performance of the original DGEMM deteriorated with the increasing number of individuals. The optimized ATLAS-DGEMM subroutine was the fastest. The performance of the optimized code for memory and loop ordering shows a similar trend to that with ATLAS-DGEMM, but is slightly slower.

Multiplication of large matrices requires an optimization to use the cache memory fully, i.e., fine tuning for specific system architectures. Simple modification to optimize indirect memory access was successful in reducing the run time. Also, a simple rearrangement of the matrix multiplication code, splitting loops and avoiding static variables within the loop seemed to allow some compilers to optimize computer operations automatically (e.g. vectorization of 'do' loops operations). Speedups were from 4 to 15 times, depending on the processor. However, code generated by the ATLAS-DGEMM subroutine ran the fastest with no additional programming as it was automatically customized for a specific processor.

Figure 3 shows results of the optimized implementation of the DGEMM subroutine with the MKL library and up to 4 processors. The speedup with four processors for 5000 genotyped animals was 3.9, close to the ideal. Actual time to create **G** for 40 K SNPs and 30 000 animals with four processors was 60 min. The operation required approximately 30 Gbytes of memory.

Creating $\mathbf{A}_{22}$ for 6500 genotyped animals using the tabular method required 311 s and 12.1 Gbyte of memory. The same computation with the Colleau (2002) required 45 s and 322 Mbytes. Whereas the tabular method requires storage for a dense matrix for all genotyped animals and their ancestors (approximately 57 000 individuals for 6500 genotyped animals), the Colleau method needs only a few vectors with the dimension equal to the number of genotyped animals. Memory requirements for the tabular method can be reduced by splitting the pedigree file into several groups, but at the cost of

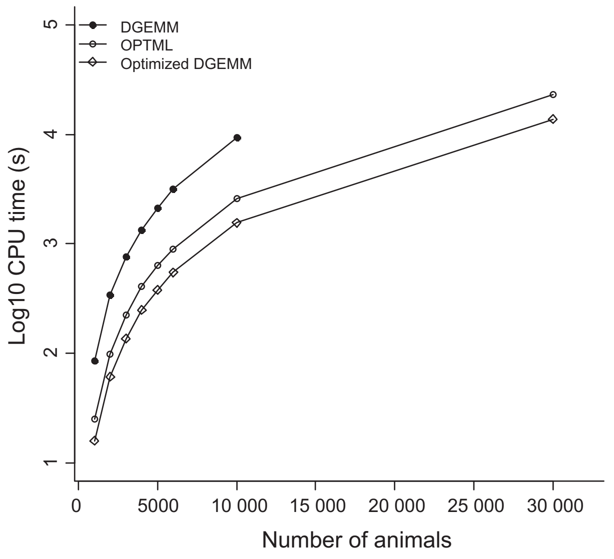

**Figure 2** Computing time using different matrix multiplication algorithms to compute G.

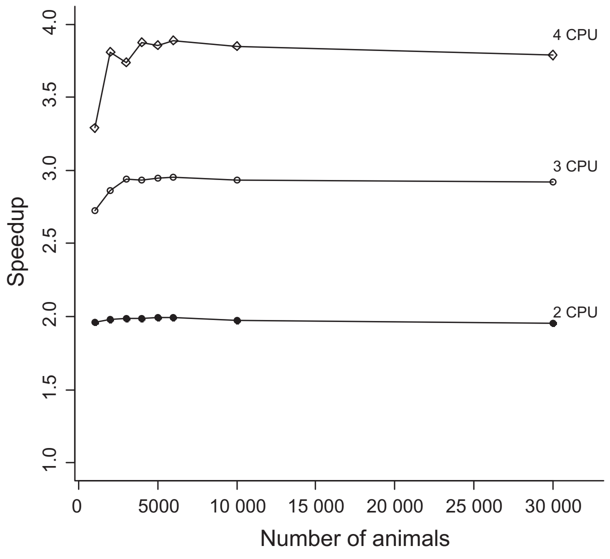

**Figure 3** Speedup for optimized DGEMM with parallel processing.

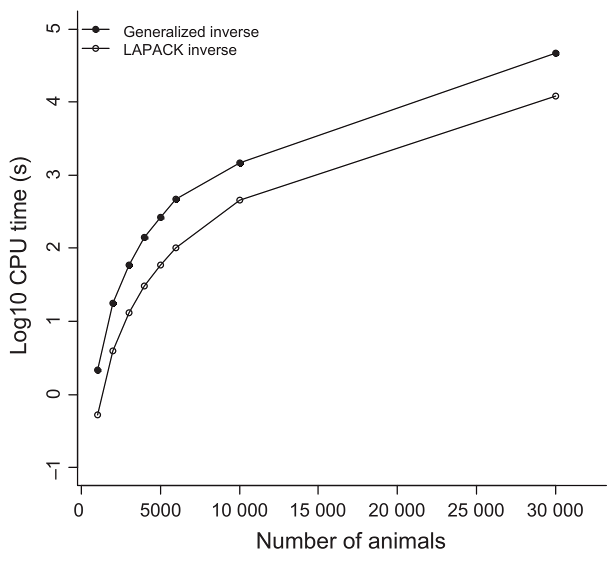

**Figure 4** Computing time of matrix inversion by generalized inverse and LAPACK subroutines.

additional computations (P. M. VanRaden, personal communication, 2009).

Computing times for inversion of different size matrices with the generalized inverse algorithm and optimized LAPACK subroutines using a single processor are shown in Figure 4. For the largest matrix (30 000 × 30 000), the inversion took approximately 13 h with the generalized inverse and 3.4 h with the optimized LAPACK. Figure 5 shows speedup of the parallel implementation using the DGEMM subroutine in the MKL. Actual time to create inverse of **G** for 40 K SNPs and 30 000 animals with four processors was <1 h.

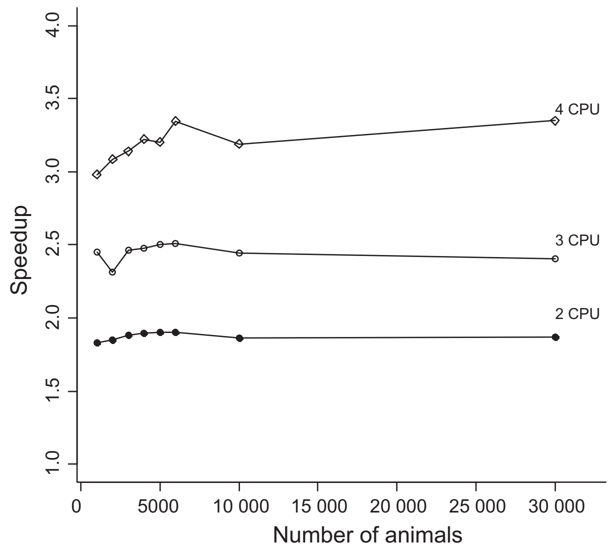

**Figure 5** Speedup for matrix inversion with optimized LAPACK subroutine using multiple processors.

Using all the optimizations for computing $\mathbf{G} = \mathbf{ZZ}'/k$, $\mathbf{A}_{22}$, $\mathbf{G}^{-1}$, and $\mathbf{A}_{22}^{-1}$ for 30 000 animals with 40 K SNPs the total time was approximately 3 h.

## Conclusion

We presented methods for efficient computation of matrices required for implementation of the single-step procedure. Optimizations were by modification of the code into simple and easy to optimize tasks by some compilers, by using existing subroutines with efficient automatic optimization provided as open source, or by commercial libraries.

## Acknowledgements

We thank P.M. VanRaden from Animal Improvement Programs Laboratory, USDA (Beltsville, MD) for providing the original software. This study was partially funded by the Holstein Association USA Inc. and by AFRI grants 2009-65205-05665 and 2010-65205-20366 from the USDA NIFA Animal Genome Program.

## References

Aguilar I., Misztal I., Johnson D.L., Legarra A., Tsuruta S., Lawlor T.J. (2010) Hot topic: a unified approach to utilize phenotypic, full pedigree, and genomic information for genetic evaluation of Holstein final score. *J. Dairy Sci.*, **93**, 743–752.

Anderson E., Bai Z., Dongarra J., Greenbaum A., McKenney A., Croz J.D., Hammerling S., Demmel J., Bischof C., Sorensen D. (1990). LAPACK: a portable linear algebra library for high-performance computers. In: Proceedings of the 1990 ACM/IEEE conference on Supercomputing, IEEE Computer Society Press, New York, New York, United States, pp. 2–11.

Astle W., Balding D. (2009) Population structure and cryptic relatedness in genetic association studies. *Statistical Science*, **24**, 451–471.

Christensen O., Lund M. (2010) Genomic prediction when some animals are not genotyped. *Genet. Sel. Evol.*, **42**, 2.

Colleau J.J. (2002) An indirect approach to the extensive calculation of relationship coefficients. *Genet. Sel. Evol.*, **34**, 409–421.

Dongarra J.J., Croz J.D., Hammarling S., Hanson R.J. (1988) An extended set of FORTRAN basic linear algebra subprograms. *ACM Trans. Math. Softw.*, **14**, 1–17.

Dongarra J., Duff I., Sorensen D., Van Der Vorst H. (1990a) Solving linear systems on vector and shared memory computers. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

Dongarra J.J., Croz J.D., Hammarling S., Duff I.S. (1990b) A set of level 3 basic linear algebra subprograms. *ACM Trans. Math. Softw.*, **16**, 1–17.

Emik L.O., Terrill C.R. (1949) Systematic procedures for calculating inbreeding coefficients. *J. Hered.*, **40**, 51–55.

Hayes B.J., Bowman P.J., Chamberlain A.J., Goddard M.E. (2009) Invited review: genomic selection in dairy cattle: progress and challenges. *J. Dairy Sci.*, **92**, 433–443.

Kang H.M., Sul J.H., Service S.K., Zaitlen N.A., Kong S.-y., Freimer N.B., Sabatti C., Eskin E. (2010) Variance component model to account for sample structure in genome-wide association studies. *Nat. Genet.*, **42**, 348–354.

Legarra A., Aguilar I., Misztal I. (2009) A relationship matrix including full pedigree and genomic information. *J. Dairy Sci.*, **92**, 4656–4663.

Misztal I., Tsuruta S., Strabel T., Auvray B., Druet T., Lee D.H. (2002). BLUPF90 and Related Programs (BGF90). In: 7th World Congress on Genetics Applied to Livestock Production. Montpellier, France, 2002.

Misztal I., Legarra A., Aguilar I. (2009) Computing procedures for genetic evaluation including phenotypic, full pedigree, and genomic information. *J. Dairy Sci.*, **92**, 4648–4655.

Nejati-Javaremi A., Smith C., Gibson J.P. (1997) Effect of total allelic relationship on accuracy of evaluation and response to selection. *J. Anim. Sci.*, **75**, 1738–1745.

Tsuruta S., Misztal I., Stranden I. (2001) Use of the preconditioned conjugate gradient algorithm as a generic solver for mixed-model equations in animal breeding applications. *J. Anim. Sci.*, **79**, 1166–1172.

VanRaden P.M. (2007) Genomic Measures of Relationship and Inbreeding. *Interbull Bull.*, **37**, 33–36.

VanRaden P.M. (2008) Efficient Methods to Compute Genomic Predictions. *J. Dairy Sci.*, **91**, 4414–4423.

VanRaden P.M., Van Tassell C.P., Wiggans G.R., Sonstegard T.S., Schnabel R.D., Taylor J.F., Schenkel F.S. (2009) Invited Review: reliability of genomic predictions for North American Holstein bulls. *J. Dairy Sci.*, **92**, 16–24.

Whaley R.C., Dongarra J.J. (1998). Automatically tuned linear algebra software. In: Proceedings of the 1998 ACM/IEEE conference on Supercomputing (CDROM). IEEE Computer Society, San Jose, CA, pp. 38.

## Appendix

Pseudo code in Fortran 95 to create pedigree-based relationship for genotyped animals. It assumes that animals are renumbered from oldest (1) to youngest (n).

```
! Create relationship matrix for genotyped animals

integer :: nGen, &           ! Number of genotyped animals

           n,&               ! Total number of animals in pedigree

           GenID(nGen),&     ! Identification of genotyped animals

           ped(n,2)          ! Pedigree matrix sire, dam

real(8) :: A22(nGen,nGen),&  ! Numerator relationship matrix for genotyped animals

           f(0:n),&          ! Inbreeding coefficients

           w(nGen),v(nGen)   ! temp vectors

….

do i = 1 , nGen

     v = 0

     v(GenId(i)) = 1

     call A_times_v(w,v,ped,f(0:n))

     A22(:,i) = w(GenId)

enddo
```

```
subroutine A_times_v(w,v,ped,f)
! Computes w=A*v using TDT'v following Colleau 2002
integer :: i, s, d, n, ped(:,:)
real (8) :: w(:), v(:), f(0:), tmp, di
real (8), allocatable :: q(:)
!
n=size(w)
allocate (q(n))
q=0
do i = n, 1, -1
   q(i) = q(i) + v(i)
   s = ped(i,1); d = ped(i,2)
   if (s /= 0) q(s) = q(s) + q(i) * 0.5
   if (d /= 0) q(d) = q(d) + q(i) * 0.5
enddo
do i = 1, n
   s = ped(i,1); d = ped(i,2)
   di = (count(ped(i,1:2) == 0) + 2) / 4.d0 - 0.25 * ( f(s) + f(d) )
   tmp = 0
   if (s /= 0) tmp = tmp + w(s)
   if (d /= 0) tmp = tmp + w(d)
   w(i) = 0.5 * tmp
   w(i) = w(i) + di *q(i)
enddo
deallocate(q)
end subroutine
```