A recursive algorithm for decomposition and creation of the inverse of the genomic relationship matrix

P. Faux,*1 N. Gengler,* and I. Misztal†
*Animal Science Unit, Gembloux Agro-Bio Tech, University of Liège, B-5030 Gembloux, Belgium
†Department of Animal and Dairy Science, University of Georgia, Athens 30602

ABSTRACT

Some genomic evaluation models require creation and inversion of a genomic relationship matrix (G). As the number of genotyped animals increases, G becomes larger and thus requires more time for inversion. A single-step genomic evaluation also requires inversion of the part of the pedigree relationship matrix for genotyped animals (A22). A strategy was developed to provide an approximation of the inverse of G (G⁻¹) that may also be applied to the inverse of A22 (A⁻22). The algorithm proceeds by creation of an incomplete Cholesky factorization (T⁻¹) of G⁻¹. For this purpose, a genomic relationship threshold determines whether 2 animals are closely related. For any animal, the sparsity pattern of the corresponding line in T⁻¹ will thus gather elements corresponding to all close relatives of that animal. Any line of T⁻¹ is filled in with resulting estimators of the least-squares regression of genomic relationships between close relatives on genomic relationships between the animal considered and those close relatives. The G⁻¹ was computed as the matrix product (T⁻¹)’D⁻¹T⁻¹, where D⁻¹ is a diagonal matrix. Then, T⁻¹G(T⁻¹)’ resulted in a new matrix that is close to diagonal and also needs to be inverted. The inverse of that matrix was approximated with the same decomposition as for approximation of the inverse of G (G⁻¹), and the procedure was repeated in successive rounds of recursion until a matrix was obtained that was close enough to diagonal to be inverted element by element. Two applications of the approximation algorithm were tested in a single-step genomic evaluation of US Holstein final score, and correlation coefficients between estimated breeding values based on either real or approximated G⁻¹ were compared. Approximations came closer to G⁻¹ as the number of recursion rounds increased. Approximations were even more accurate and expected to be faster for A22. Timesaving strategies are needed to reduce the computing time required for the algorithm.

Key words: genomic selection, relationship matrix, matrix inversion, incomplete Cholesky factorization

INTRODUCTION

Meuwissen et al. (2001) proposed a method to predict breeding values that includes molecular information from dense panel chips that cover the whole genome. This method, called genomic prediction, relies on the assumption that markers are expected to be in linkage disequilibrium with potential QTL. Among different methods proposed to implement genomic prediction, VanRaden (2007) suggested that a genomic relationship matrix (G) could be computed based on molecular knowledge of alleles shared between individuals, and that mixed models using such a matrix instead of a pedigree-based matrix (A) could predict genetic effects more accurately than those using A. The main structural difference between A and G is that individuals assumed to be unrelated based on pedigree almost always share fractions of their genomes. Therefore, G is expected to be dense, whereas A may be sparse, and use of G in genomic BLUP (GBLUP) consequently requires inversion of a dense matrix.

Legarra et al. (2009) developed a method that includes pedigree, phenotypic, and genomic information in one step, which was also shown by Christensen and Lund (2010) and mentioned by Bömcke et al. (2011). That method requires inversion of G as well as inversion of the part of A that represents pedigree-based relationships among genotyped animals (A22). Although A may be sparse and able to be inverted using the rules of Henderson (1976) and Quaas (1976), A22 is expected to be less sparse and cannot be inverted using those rules because it contains relationships from a nongenotyped ancestor. The G and A22 matrices can be created and inverted by standard procedures such as those in the program PREGSF90 (Aguilar et al., 2011) by using a package of linear algebra kernels (Basic Linear Algebra Subprograms; Lawson et al., 1979) that are...
able to multiply matrices efficiently. Using PREGSF90, Aguilar et al. (2011) showed that creation and inversion of $A_{22}$ and $G$ for 30,000 animals with 40,000 SNP required approximately 3 h. However, their computations increase cubically with the number of animals and are unsuitable for very large numbers of genotypes.

The inverse of $A$ ($A^{-1}$) can be calculated quickly and with minimum storage because it is based on a recursion formula involving only 3 individuals: an animal and its parents. For direct inversion of $G$, genomically enhanced EBV (GEBV) of a genotyped animal is assumed dependent on all other genotyped animals. The first objective of this study was to develop and evaluate a recursion formula for the inverse of $G$ ($G^{-1}$) that includes only a fraction of genotyped animals. The second objective was to determine whether such a formula is also suitable for creation of the inverse of $A_{22}$ ($A_{22}^{-1}$).

**MATERIALS AND METHODS**

**Approximation of Inverse of $G$ and $A_{22}$**

**State of the Art.** Development of approximations will be based on root-free Cholesky factorization. If the genomic relationship matrix $G$ is a symmetric positive definite matrix, then $G$ is invertible and can be factorized as $G = LL'$. The decomposition of its inverse is thus $(L^{-1})L^{-1}$, which is equal, if $L = T\tilde{D}$, to the following root-free Cholesky factorization of $G^{-1}$:

$$G^{-1} = (T^{-1})\tilde{D}^{-1}T^{-1}. \quad [1]$$

Note that root-free factorization means that $T^{-1}$ always has 1 on the diagonal.

We will focus hereafter on approximating the inverse of the root-free Cholesky factor ($T^{-1}$) rather than the root-free Cholesky factor itself ($T$).

Even if less popular than approximations of the Cholesky factor, sparse approximations of the inverse of the Cholesky factor of a nonsingular coefficient matrix, for instance, $M$, are frequently used for computation of preconditioners for conjugate gradient calculations of linear systems involving $M$. In a comparative study, Benzi and Tuma (1999) identify 2 main approaches of computing such sparse approximations. In the first approach (to which our algorithm belongs), the sparse approximation of the inverse of the Cholesky factor is directly computed from $M$. Information about triangular factors of $M$ is not required. Among others with the same approach, a method by Kolotilina and Yeremin (1993) might be noticed. This method computes a factorized sparse approximate inverse (FSAI) by minimizing the Frobenius norm between the approximate and the real triangular factor and the approximation of its inverse, as $\|I - PQ_1\|_F$, where $I$ is an identity matrix, $P$ is the sparse approximation of $Q^{-1}$, and $Q$ is the Cholesky factor of a symmetric $M$. In contrast, the second approach gathers methods that require an incomplete factorization of $M$ and that will use this incomplete factorization to obtain a sparse approximation of the inverse of the Cholesky factor. Nevertheless, our aim is not to use a sparse approximation of the inverse of the triangular factor for preconditioning, but is to use that sparse approximation for approximating the inverse of the matrix.

**Algorithm for Approximation of Inverse of Cholesky Factor.** Based on equation [1], we propose an approximation $\tilde{G}^{-1}$ of $G^{-1}$, expressed as

$$\tilde{G}^{-1} = \tilde{T}\tilde{D}^{-1}\tilde{T}, \quad [2]$$

where $\tilde{T}$ is an approximation of $T^{-1}$, and $\tilde{D}^{-1}$ is an approximation of the inverse of $D$.

The method that we developed to create a sparse $\tilde{T}$ will be illustrated through an example pedigree (Figure 1) for 4 animals ($s$, $d$, $j$, and $i$). Animals $s$ and $d$ are parents of $i$, and animal $j$ represents any other animal older than $i$ and related to $i$ through ancestors of $s$ and $d$. Additional ancestors that explain the relationships among $j$, $s$, and $d$ are omitted. For the example pedigree, assume that genomic relationships ($g$) among animals are as follows:

$$
\begin{bmatrix}
g_{ss} & g_{sd} & g_{sj} & g_{si} 
g_{ds} & g_{dd} & g_{dj} & g_{di} 
g_{js} & g_{jd} & g_{jj} & g_{ji} 
g_{is} & g_{id} & g_{ij} & g_{ii}
\end{bmatrix} =
\begin{bmatrix}
1.13 & 0.04 & 0.17 & 0.58 
0.04 & 1.02 & 0.32 & 0.47 
0.17 & 0.32 & 0.98 & 0.22 
0.58 & 0.47 & 0.22 & 1.09
\end{bmatrix}.
$$

Genomic relationships stretch across the whole pedigree and potentially across breeds. Therefore, we will introduce the concept of close family where close family ($\Omega_i$) of any animal $i$ may be defined by a genomic relationship threshold $p$ as

$$\Omega_i = \{j : j < i, g_{ij} \geq p\}. \quad [3]$$

Note that, due to the condition on $j$, the close family is restricted to animals older than $i$. In the current example, we have defined $p$ as equal to 0.15.

The algorithm will fill in $T$ as follows, for any animal $i$ in pedigree:

(i) select all animals in the close-family of $i$; that is, animals $j$ as defined by expression [3];
Figure 1. Pedigree relationships between 4 animals $s$, $d$, $j$, and $i$, where $s$ and $d$ are parents of $i$, and $j$ is any animal older than $i$ and related to both $s$ and $d$ through ancestors.

(ii) perform the regression of relationships between those animals on relationships that they share with animal $i$;

(iii) fill in positions in $\hat{T}$ that correspond to animals selected in $\Omega$, with opposite of solutions and set diagonal element of $\hat{T}$ to 1.

For the example mentioned before, we run here the algorithm for the last animal in pedigree, $i$:

(i) $\Omega_i = \{s, d, j\}$, as all animals in pedigree have a genomic relationship, with $i$ greater than $p = 0.15$;

(ii) the regression to be performed is here illustrated by the example

\[
\begin{pmatrix}
g_{ss} & g_{sd} & g_{sj} \\
g_{ds} & g_{dd} & g_{dj} \\
g_{is} & g_{jd} & g_{ii}
\end{pmatrix} \cdot \hat{\beta} + \hat{\varepsilon} = \begin{pmatrix} g_s \\ g_d \\ g_i \end{pmatrix}
\]

\[
\begin{pmatrix}
1.13 & 0.04 & 0.17 \\
0.04 & 1.02 & 0.32 \\
0.17 & 0.32 & 0.98
\end{pmatrix} \cdot \hat{\beta} + \hat{\varepsilon} = \begin{pmatrix} 0.58 \\ 0.47 \\ 0.22 \end{pmatrix}
\]

where $\hat{\beta}$ is a regression coefficients vector and $\hat{\varepsilon}$ is a vector of residuals; by ordinary least-squares, solutions $\hat{\beta}$ of this regression are

\[
\begin{pmatrix}
0.50 & 0.44 & -0.01
\end{pmatrix}
\]

and residuals $\hat{\varepsilon}$ are \([0 \ 0 -0.00]^T\);

(iii) $\hat{T}$ is filled in with opposite of these solutions and its diagonal elements are set to 1, which gives, for the example, the final $\hat{T}$:

\[
\hat{T} = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
-0.14 & -0.31 & 1 & 0 \\
-0.50 & -0.44 & 0.01 & 1
\end{bmatrix}
\]

As can be seen, $\hat{T}$ is filled in only in positions where the corresponding values in the lower triangular part of $G$ are greater than $p$. This threshold was here fixed to 0.15 and allows us to control the sparsity of $\hat{T}$. This sparse approximation has to be compared with the matrix that it aims to approximate (see equation [1]), which is, by complete Cholesky factorization and inversion of the triangular factor, as follows:

\[
T^{-1} = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
-0.04 & 1 & 0 & 0 \\
-0.14 & -0.31 & 1 & 0 \\
-0.50 & -0.44 & 0.01 & 1
\end{bmatrix}
\]

The next step is to compute $D$ (see equation [1]) and to approximate its inverse ($D^{-1}$).

**Recursive Formula and Approximation of $D^{-1}$**.

A quick rearrangement of equation [1] gives

\[
D = T^{-1}G(T^{-1})'.
\]

Replacing $T^{-1}$ by its approximation $\hat{T}$ in the formula above allows us to compute a $D$ that is no longer diagonal, but close to diagonal. For the current example, $D$ would be

\[
D = \begin{bmatrix}
1.13 & 0.04 & 0 & 0 \\
0.04 & 1.02 & 0 & 0 \\
0 & 0 & 0.86 & 0 \\
0 & 0 & 0 & 0.59
\end{bmatrix}
\]

Thus, $D$ approaches a diagonal matrix. Nevertheless, $D$ has to be inverted or its inverse approximated ($D^{-1}$).

If $D$ is almost a diagonal matrix, then inversion of only its diagonal elements may be a good approximation of its inverse. However, if $D$ is not close enough to being diagonal, the algorithm given above can be used to approximate $D^{-1}$ instead of $G^{-1}$.

Therefore, by its recursive use, the algorithm leads to the following $D^{-1}$ in round n. Similar to approximation in equation [2],

\[
\hat{D}_{n+1}^{-1} = T_n\hat{D}_n^{-1}T_n.
\]

Equation [6] involves choice of the new parameter \( p \), construction of the new matrix \( \mathbf{T}_n \), and computation of the new matrix \( \mathbf{D}_n = \mathbf{T}_n \mathbf{D}_{n-1}^{-1} \mathbf{T}_n \).

After \( t \) rounds of recursion, when the off-diagonal elements of \( \mathbf{D}_n \) are considered to be small enough,

\[
\mathbf{G}_t^{-1} = \mathbf{T}_t \mathbf{D}_t^{-1} \mathbf{T}_t, \tag{7}
\]

where \( \mathbf{D}_t^{-1} \) is the matrix formed by the inverse of the diagonal elements of \( \mathbf{D}_t \). If \( \mathbf{T}_j = \mathbf{T}_t \mathbf{D}_t^{-1} \mathbf{T}_t \), then equation [7] becomes

\[
\mathbf{G}_t^{-1} = \mathbf{T}_j \mathbf{D}_t^{-1} \mathbf{T}_j. \tag{8}
\]

**Link with Inversion of \( \mathbf{A} \)**

The pedigree-based relationship matrix \( \mathbf{A} \) contains relationships \( a \) that are obtained iteratively with the rule

\[
a_{ij} = 0.5a_{ij} + 0.5a_{dj}, \tag{9}
\]

where \( i \) and \( j \) are 2 animals (with \( j \) older than \( i \)), and \( s \) and \( d \) are the parents of \( i \). Therefore, \( a_{ij} \) represents the pedigree-based relationship between \( i \) and \( j \); \( a_{ij} \) the relationship between \( s \) and \( j \); and \( a_{dj} \), the relationship between \( d \) and \( j \).

For the example pedigree (Figure 1), the additive relationships among the 4 animals \( s \), \( d \), \( j \), and \( i \) may be

\[
\begin{bmatrix}
  a_{ss} & a_{sd} & a_{sj} & a_{si} \\
  a_{ds} & a_{dd} & a_{dj} & a_{di} \\
  a_{js} & a_{jd} & a_{jj} & a_{ji} \\
  a_{as} & a_{ad} & a_{aj} & a_{ai}
\end{bmatrix} =
\begin{bmatrix}
  1.00 & 0.00 & 0.20 & 0.50 \\
  0.00 & 1.00 & 0.30 & 0.50 \\
  0.20 & 0.30 & 1.00 & 0.25 \\
  0.50 & 0.50 & 0.25 & 1.00
\end{bmatrix}. \tag{10}
\]

Applying the algorithm of construction of \( \mathbf{T} \) to the case of \( \mathbf{A} \), with the same threshold \( p \) (\( p = 0.15 \)), leads, for the fourth animal (\( i \)), to the regression in [10]:

\[
\begin{bmatrix}
  a_{ss} & a_{sd} & a_{si} \\
  a_{ds} & a_{dd} & a_{di} \\
  a_{js} & a_{jd} & a_{ji} \\
  a_{as} & a_{ad} & a_{ai}
\end{bmatrix} = \begin{bmatrix}
  a_{ss} & a_{sd} & a_{sj} & a_{si} \\
  a_{ds} & a_{dd} & a_{dj} & a_{di} \\
  a_{js} & a_{jd} & a_{jj} & a_{ji} \\
  a_{as} & a_{ad} & a_{aj} & a_{ai}
\end{bmatrix} \beta + \epsilon
\]

\[
\Leftrightarrow \begin{bmatrix}
  0.50 & 0.00 & 0.00 & 0.20 \\
  0.00 & 1.00 & 0.00 & 0.30 \\
  0.20 & 0.30 & 1.00 & 0.00
\end{bmatrix} = \begin{bmatrix}
  1.00 & 0.00 & 0.20 & 0.50 \\
  0.00 & 1.00 & 0.30 & 0.50 \\
  0.20 & 0.30 & 1.00 & 0.25
\end{bmatrix} \beta + \epsilon, \tag{11}
\]

which, due to the iterative rule of computation of pedigree-based relationships (see equation [9]), has trivial solutions. Whatever \( j \) is, solutions will always be \( \beta_s = 0.5, \beta_d = 0.5, \beta_j = 0; \) that is, always zeros for all animals except for parents, for which the solution is 0.5.

Filling in \( \mathbf{T} \) with opposite of these solutions is actually \( \mathbf{A}^{-1} \), used for direct computation of \( \mathbf{A}^{-1} \):

\[
\mathbf{A}^{-1} = (\mathbf{T}_n^{-1})^\dagger \mathbf{D}^{-1} \mathbf{T}_n^{-1}. \tag{12}
\]

The equivalence is easily shown when replacing unknowns by their trivial solutions (0.5) in equation [9]:

\[
\begin{bmatrix}
  a_{ss} & a_{sd} & a_{sj} \\
  a_{ds} & a_{dd} & a_{dj} \\
  a_{js} & a_{jd} & a_{jj} \\
  a_{as} & a_{ad} & a_{aj}
\end{bmatrix} =
\begin{bmatrix}
  0.50 & 0.00 & 0.00 \\
  0.00 & 1.00 & 0.00 \\
  0.00 & 0.00 & 0.50 \\
  0.00 & 0.00 & 0.00
\end{bmatrix} + \mathbf{A}^{-1} = \mathbf{A}^{-1} = \mathbf{e}, \tag{13}
\]

and rearranging this last equation, which gives

\[
\begin{bmatrix}
  -0.50 & -0.50 & 0 \\
  0.50 & 0.50 & 0 \\
  0.00 & 0.00 & 0
\end{bmatrix} = \mathbf{A}^{-1} = \mathbf{e} = [0 \ 0 \ 0]. \tag{14}
\]

The product in equation [13] is the product \( \mathbf{T} \mathbf{A} \) restricted to line \( i \) of \( \mathbf{T} \) and columns \( s, d, \) and \( j \) of \( \mathbf{A} \). As constructed above, \( \mathbf{T} \) is thus always equal to the highly sparse \( \mathbf{D}^{-1} \) of equation [12]. Furthermore, residuals of estimation are always zero (equation [14]) and, in respect to equation [3], they correspond to off-diagonals of \( \mathbf{D} \), which ensures that \( \mathbf{D} \) is always diagonal, and thus easily inverted.

What our algorithm performs on \( \mathbf{G} \) is thus inspired by what is done in the direct computation of \( \mathbf{A}^{-1} \), but extended, in our case, to all animals closely related to a given animal, rather than only to his parents.

**Applications of Algorithm**

Two computing bottlenecks occur: construction of each \( \mathbf{T} \) and matrix multiplications. In both cases, the number of estimates \( k \) for each line of \( \mathbf{T} \) (i.e., the value chosen for \( p \)) is critical. A lower \( p \) results in longer time for ordinary least squares estimation and a less sparse \( \mathbf{T} \). Consequently, 2 different applications of the algorithm were examined. For the first application, \( k \)
Tests of Algorithm

**Computation of G and A**

The genomic relationship matrix was here computed as in VanRaden (2008):

\[ G = \frac{Z Z'}{2 \sum_{i=1}^{m} p_i (1 - p_i)} \]

where \( Z \) is a matrix of \( n \) rows (\( n = \) number of genotyped animals) and \( m \) columns (\( m = \) total number of markers) that contains genotypes centered around the frequency \( p_i \) of the second allele at locus \( i \). As discussed by Forni et al. (2011), matrix compatibility can be obtained by using observed allelic frequency (rather than fixed allelic frequency) and rescaling the genomic relationship matrix so that the mean for diagonal elements is 1; that is, on the same scale as diagonal elements of \( A_{22} \). In addition, off-diagonal elements of \( G \) were also scaled to be comparable with off-diagonal elements of \( A_{22} \). All computations were performed using PREGSF90, which also created \( A_{22} \) using algorithm by Colleau (2002).

**Phenotypic Records, Pedigree, and Genotypes.**

The algorithm was tested using data for the official May 2009 US Holstein genetic evaluation for final score, which were provided by Holstein Association USA Inc. (Brattleboro, VT). A total of 10,553,183 final score records from 6,296,878 cows were available, as well as 9,120,198 pedigree records. In total, 6,931 bulls had been genotyped using the BovineSNP50 BeadChip (Illumina Inc., San Diego, CA). After removal of uninformative and low-quality SNP, 38,416 SNP were used to estimate genomic relationship coefficients. For computational ease, a reduced sample of 1,718 genotyped animals and all their genotyped ancestors.

Evaluations were calculated with an animal model that included fixed effects for management group (herd-year-classifier), age group by classification year, and lactation stage by classification year as described by Tsuruta et al. (2004). Variance components were those used for national evaluation of final score. Heritability was equal to 0.35 and repeatability was equal to 0.67. Evaluations were calculated using approximations of \( G^{-1} \) and \( A^{-1} \) from both applications of the algorithm.

**Tests on Approximations.**

Quality of approximation of \( G^{-1} \) was tested by mean square difference (MSD) between \( G^{-1} \) and \( \tilde{G}^{-1} \) calculated as a weighted sum of square differences (here, \( n \) denotes the size of the matrix):

\[ MSD = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} (G^{-1}_{ij} - \tilde{G}^{-1}_{ij})^2}{n^2}. \]

Moreover, the degree of sparsity of \( \tilde{T}_f \) that may be returned after any number of rounds of recursion was measured as the percentage of elements that equaled zero in the lower off-diagonal part of the matrix. Because multiplications of \( \tilde{T}_f \) provide the inverse approximation, a high percentage of elements in \( \tilde{T}_f \) equal to zero (i.e., a high degree of sparsity) should result in rapid computational time for the algorithm. Corresponding MSD and percentage of elements of \( \tilde{T}_f \) that equaled zero were also calculated for \( A_{22}^{-1} \) and its approximation \( \tilde{A}_{22}^{-1} \).

Even if differences between \( G^{-1} \) and \( \tilde{G}^{-1} \) exist, they might not be relevant for genomic predictions. Therefore, each \( \tilde{G}^{-1} \) was used in the single-step procedure of Misztal et al. (2009) to compute approximated GEBV for final score, including phenotypic, pedigree, and genomic information for 1,718 genotyped animals. Linear regressions of GEBV computed using approximated inverse of \( G \) on GEBV computed with real \( G^{-1} \) were estimated. Correlations between both sets of GEBV (computed using either real or approximated inverse) and standard deviations of GEBV computed using approximated inverse were also estimated.

**RESULTS AND DISCUSSION**

**Approximation of G**

The MSD and the percentage of elements of the lower triangular part of \( T_f \) that equaled zero are shown in Table 1 for the first application of the algorithm (5 rounds of recursion with a smaller \( p \) at each round) and in Table 2 for the second application of the algorithm (8 rounds of recursion with a maximum \( k \) of 50 for all rounds). As for any approximation, MSD between \( G^{-1} \) and \( \tilde{G}^{-1} \) was always higher than MSD between \( G^{-1} \) and \( A_{22}^{-1} \) (81.49 × 10⁻⁴), which suggests that any \( \tilde{G}^{-1} \) would be intermediate to \( A_{22}^{-1} \) and \( G^{-1} \). With more rounds of recursion, \( \tilde{G}^{-1} \) became closer to \( G^{-1} \), whereas \( \tilde{T}_f \) became denser. The close relationships among the Holstein bulls may partly explain the loss of sparsity. A population that included different breeds (or even different
lines within breed) would be expected to have different results. However, in such a multi-breed situation, our method could have an additional advantage because animals from different breeds would not be closely related (i.e., they would share genomic relationships lower than the threshold \( p \)), and would therefore generate an inverted matrix that separates breeds.

### Estimation of GEBV with Approximate \( G^{-1} \)

Slopes of linear regression of GEBV obtained using approximated inverse of \( G \) (denoted hereafter as “approximated GEBV”) on GEBV obtained using real inverse of \( G \) (denoted hereafter as “real GEBV”), correlation coefficients between both sets of GEBV and standard deviations of approximated GEBV are in Table 3 for the first algorithm application and in Table 4 for the second application. Regardless of the algorithm application or the rounds of recursions, all slopes were higher than that for linear regression of EBV calculated without genomic information (hereafter denoted as “traditional EBV”) on real GEBV (0.59). All correlation coefficients were also larger than the correlation coefficient between traditional EBV and real GEBV (0.76). The standard deviations of all approximated GEBV were between the standard deviation of traditional EBV (10.69) and the standard deviation of real GEBV (13.77). The first application of the algorithm needed only 3 rounds to achieve results similar to those obtained after 7 rounds of the second application. For both applications, a very similar relationship was observed between sparsity of the approximated \( G^{-1} \) and the correlation between GEBV: a decreasing sparsity in the factorization of \( G^{-1} \) leads to more similar GEBV. A strong increase in correlation between approximated and real GEBV was also observed in the first rounds of recursion for both applications, indicating that an important part of the genomic relationships was recovered.

#### Approximation of \( \tilde{A}_{22}^{-1} \)

All \( \tilde{A}_{22}^{-1} \) were calculated with the same sequence of thresholds \( p \) used in the first application of the algorithm to calculate \( G^{-1} \) (Table 1) and the same value of \( k \) as in the second application (Table 2) for \( \tilde{G}^{-1} \). For each application, after the same number of recursion rounds, MSD were smaller for \( \tilde{A}_{22}^{-1} \) than for \( \tilde{G}^{-1} \), and \( \tilde{T}_f \) were much sparser (Table 5 for the first algorithm application and Table 6 for the second application). Furthermore, the nonzero elements of \( \tilde{T}_f \) tended to have a higher proportion of small elements (<0.01) when calculating \( \tilde{A}_{22}^{-1} \) compared with \( \tilde{G}^{-1} \) (Figure 2 for the first algorithm application; Figure 3 for the second application). Computational enhancement by removal of the smallest coefficients should thus be possible for calculating \( \tilde{A}_{22}^{-1} \). The results suggest that the algorithm is suitable for inversion of \( A_{22} \).

<table>
<thead>
<tr>
<th>Recursion round</th>
<th>( p )</th>
<th>MSD</th>
<th>( \theta ) (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.210</td>
<td>66.53 \times 10^{-4}</td>
<td>92.35</td>
</tr>
<tr>
<td>2</td>
<td>0.017</td>
<td>34.87 \times 10^{-4}</td>
<td>10.78</td>
</tr>
<tr>
<td>3</td>
<td>0.009</td>
<td>16.81 \times 10^{-4}</td>
<td>2.28</td>
</tr>
<tr>
<td>4</td>
<td>0.005</td>
<td>5.97 \times 10^{-4}</td>
<td>0.86</td>
</tr>
<tr>
<td>5</td>
<td>0.003</td>
<td>1.82 \times 10^{-4}</td>
<td>0.49</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Recursion round</th>
<th>( p )</th>
<th>Slope</th>
<th>( r )</th>
<th>( s )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.210</td>
<td>0.68</td>
<td>0.79</td>
<td>11.81</td>
</tr>
<tr>
<td>2</td>
<td>0.017</td>
<td>0.86</td>
<td>0.91</td>
<td>13.04</td>
</tr>
<tr>
<td>3</td>
<td>0.009</td>
<td>0.92</td>
<td>0.97</td>
<td>13.54</td>
</tr>
<tr>
<td>4</td>
<td>0.005</td>
<td>0.97</td>
<td>0.99</td>
<td>13.54</td>
</tr>
<tr>
<td>5</td>
<td>0.003</td>
<td>0.99</td>
<td>&gt;0.99</td>
<td>13.66</td>
</tr>
</tbody>
</table>

### Table 3. Slopes of linear regression of genomically enhanced EBV (GEBV) based on approximated genomic relationship matrix (approximated GEBV) from 5 rounds of recursion of an approximation algorithm that defines the number of estimates for each line of the triangular matrix by the genomic relationship threshold \( p \) on GEBV based on the actual genomic relationship matrix (real GEBV), coefficient of correlation \( r \) between approximated and real GEBV, and standard deviations of approximated GEBV \( s \) for 1,718 genotyped US Holsteins evaluated for final score in May 2009

<table>
<thead>
<tr>
<th>Recursion round</th>
<th>( p )</th>
<th>Slope</th>
<th>( r )</th>
<th>( s )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.210</td>
<td>0.68</td>
<td>0.79</td>
<td>11.81</td>
</tr>
<tr>
<td>2</td>
<td>0.017</td>
<td>0.86</td>
<td>0.91</td>
<td>13.04</td>
</tr>
<tr>
<td>3</td>
<td>0.009</td>
<td>0.92</td>
<td>0.97</td>
<td>13.54</td>
</tr>
<tr>
<td>4</td>
<td>0.005</td>
<td>0.97</td>
<td>0.99</td>
<td>13.54</td>
</tr>
<tr>
<td>5</td>
<td>0.003</td>
<td>0.99</td>
<td>&gt;0.99</td>
<td>13.66</td>
</tr>
</tbody>
</table>
Table 4. Slopes of linear regression of genomically enhanced EBV (GEBV) based on approximated genomic relationship matrix (approximated GEBV) from 8 rounds of recursion of an approximation algorithm that defines the maximum number of estimates to be 50 for all rounds on GEBV based on the actual genomic relationship matrix (real GEBV), coefficient of correlation (r) between approximated and real GEBV, and standard deviations of approximated GEBV (s) for 1,718 genotyped US Holsteins evaluated for final score in May 2009.

<table>
<thead>
<tr>
<th>Recursion round</th>
<th>Slope</th>
<th>r</th>
<th>s</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.75</td>
<td>0.77</td>
<td>13.39</td>
</tr>
<tr>
<td>2</td>
<td>0.84</td>
<td>0.87</td>
<td>13.36</td>
</tr>
<tr>
<td>3</td>
<td>0.87</td>
<td>0.90</td>
<td>13.30</td>
</tr>
<tr>
<td>4</td>
<td>0.89</td>
<td>0.93</td>
<td>13.23</td>
</tr>
<tr>
<td>5</td>
<td>0.92</td>
<td>0.95</td>
<td>13.41</td>
</tr>
<tr>
<td>6</td>
<td>0.94</td>
<td>0.96</td>
<td>13.47</td>
</tr>
<tr>
<td>7</td>
<td>0.95</td>
<td>0.97</td>
<td>13.55</td>
</tr>
<tr>
<td>8</td>
<td>0.96</td>
<td>0.98</td>
<td>13.59</td>
</tr>
</tbody>
</table>

### Possible Improvements

The algorithm described provides a new insight on inversion of G. Each round of recursion returns an approximation that is better than the previous one, with convergence achieved in a few rounds. Consequently, the main issue for the proposed algorithm is the computing time. Currently, no efforts have been made to study optimization, but reduction of computing time could be achieved in 3 ways. First, matrix computations could be optimized, for example, as in Aguilar et al. (2011). Unfortunately, this algorithm would still have a cubic cost. Second, the number of animals selected for ordinary least squares could be decreased without compromising approximation accuracy. Hayes et al. (2009) stated that the gain in accuracy by including different breeds in the same relationship matrix is low or close to zero. In addition, Muir (2007) found by simulation that genomic predictive ability strongly decays with generations under strong selection (compared with random selection). That finding was corroborated by practical studies in layer chickens (Wolc et al., 2011). Therefore, a recursion algorithm may eliminate animals that are more than 1 to 2 generations apart or are from another line or breed. With actual data sets, the number of such animals may be approximately constant even with increasing numbers of genotypes. Therefore, time for computation of any T would be equal regardless of the number of genotypes. Besides, if only genomic relationships between closely related animals are sufficient to properly predict GEBV, it suggests that the whole creation of G might be avoided. The storage in RAM of the whole G would thus not be required and, in the frame of our algorithm, elements of D would be kept in memory only if they exceeded the next threshold p.

The third way to reduce computing time is related to the selection of regression variables for ordinary least squares. Two methods to select animals were presented: selection of all animals based on a genomic relationship threshold (first algorithm application) and selection of a specified number of closest-related animals (second algorithm application). For both selection methods, genomic (or A22) relationships for a given animal are defined by genomic (or A22) relationships among all its relatives. However, some relationships with relatives could be the same for 2 closely related animals; that is, the incidence matrix in the ordinary least-squares equations (equation [4]) of 2 animals could contain identical blocks. Thus, some matrix manipulation could easily lead to solutions for the second animal based on solutions for the first animal. This method of reducing computing time achieves some of the advantage realized from decreasing the number of animals selected for ordinary least squares. Moreover, animals are so far selected based only on their closeness (using a thresh-
old p) to deduct linear combinations between their relationships. This criterion (closeness) could be enhanced; as for the case of A, some close animals (for instance, half-sibs) might not be needed, whereas close parents of close animals would be. If x is an animal of interest for estimation of genomic relationships of a given animal y closely related to him, then, z, parent of x though not related to y, might be helpful for estimation as it would explain not the similarity but the difference between x and y.

The development of this approximation algorithm was based on the assumption that using a complete G
for genomic evaluation is appropriate, and algorithm approximations were compared with $G$ that links all genotyped animals. However, the inclusion of some distant relationships in $G$ could be detrimental to evaluation accuracy. In such a case, the approximated $G$, which assumes contributions of closely related individuals from recent generations and from the same line, would result in higher accuracy. For example, if a few breeds were evaluated together with no predictability across breeds, $G^{-1}$ from regular algorithms would contain nonzero off-diagonal elements across breeds. Those elements, which should be zero, would be a source of “noise” in GEBV. By selecting only closely related animals, the recursive approximation algorithm would set those elements to zero automatically and could be more beneficial. However, additional research is needed to clarify this point.

**CONCLUSIONS**

This recursive algorithm is powerful enough to approximate the inverse of a matrix such as $G$ or $A_{22}$. As the number of rounds of recursion increases, the inverse approximation becomes closer to the real inverse. The coefficient of correlation between approximated and real GEBV showed a strong increase in the first recursion rounds, indicating that few rounds may be enough to recover important genomic relationships. Currently used applications of the algorithm can be optimized to achieve efficient selection of closely related animals. The algorithm may also be particularly suitable in the case of the inversion of $A_{22}$ as it achieves a highly sparse factorization of this matrix within a few rounds of recursion. As quality of approximation depends on the number of rounds of recursion, the computing time required by each additional round is crucial and has to be optimized. As shown in before, several computational improvements that could be developed in the future might reduce the computational time required.

**ACKNOWLEDGMENTS**

The current project (NextGenGES-PHD-09-118) is supported by the National Research Fund, Luxembourg, as part of a private public partnership between CONVIS (Etterbuck, Luxembourg) and Gembloux Agro-Bio Tech of the University of Liège. This project was partially supported by AFRI grants 2009-65205-05665 and 2010-65205-20366 from USDA NIFA (Washington, DC). The authors gratefully acknowledge the financial support (project D31-1233/S1) of the Ministry of Agriculture of the Walloon Region of Belgium. Nicolas Gengler, who was Senior Research Associate of the National Fund for Scientific Research (FNRS) until December 31, 2011, acknowledges his support. The authors thank Holstein Association USA Inc. (Brattleboro, VT) and the Cooperative Dairy DNA Repository (Beltsville, MD) for providing genotypic data. Editorial help by Suzanne Hubbard is gratefully acknowledged. Helpful comments by reviewers are acknowledged.

**REFERENCES**


APPENDIX

Application 1

The genomic matrix $G$ first is ordered from oldest to youngest animal. For any row $i$ from 2 to $n$, the algorithm proceeds as follows:

1. Select individuals older than animal $i$ and with a genomic relationship with animal $i$ that is larger than the parameter $p$ defined for this recursion round. If none can be selected under that condition, take all individuals older than animal $i$ in round 1 or ignore the line (i.e., none is selected) in the next rounds.
2. Perform ordinary least squares regression of the genomic relationships among those animals on their genomic relationships with animal $i$.
3. Update the lower triangular matrix $T$ with the regression estimates on the off-diagonals and 1 on the diagonal.
4. Then, as in equation [6], $T_i^r G T_i^r$ in round 1 returns $D_i$, and $T_i D_{r-1} T_i^r$ in each of the next rounds returns $D_r$.
5. Finally, $T_i$ in round 1 or $T_i T_i^r$ in each of the next rounds updates $T_i$.

After $t$ rounds, a last product returns $\hat{G}_t^{-1}$ using equation [8], where $\hat{D}_t^{-1}$ is made up of diagonal elements of $D_t$. Steps 1 and 3 occur at each round but are different for round 1 and successive rounds. Steps 2 and 4 occur at each round. Step 5 does not occur for round 1.

The assignation of a value to threshold $p$ in step 1 has been made arbitrarily. This value nonetheless depends on the distribution of off-diagonals of the matrix, for which the inverse has to be approximated and may be deduced from this distribution.

Application 2

Only step 1 of application 1 is changed for application 2. Instead of selecting all animals that are older than animal $i$ and have a genomic relationship with animal $i$ that is greater than parameter $p$, a maximum of $k$ animals is selected. If $i$ is $\leq (k + 1)$, all animals are selected, and if $i$ is $> (k + 1)$, the $k$ animals with the largest genomic relationships with animal $i$ are selected. This modification avoids regression with more than $k$ estimators. In addition, each $T$ created is sparser, and the increment is equal from each round to the next ($k$ estimators are added).