Fernando *et al. Genet Sel Evol (2016) 48:96*
DOI 10.1186/s12711-016-0273-2

Genetics Selection Evolution

RESEARCH ARTICLE
Open Access

Computational strategies for alternative single-step Bayesian regression models with large numbers of genotyped and non-genotyped animals

Rohan L. Fernando[1*], Hao Cheng[1], Bruce L. Golden[2] and Dorian J. Garrick[1,3]

Abstract

Background: Two types of models have been used for single-step genomic prediction and genome-wide association studies that include phenotypes from both genotyped animals and their non-genotyped relatives. The two types are breeding value models (BVM) that fit breeding values explicitly and marker effects models (MEM) that express the breeding values in terms of the effects of observed or imputed genotypes. MEM can accommodate a wider class of analyses, including variable selection or mixture model analyses. The order of the equations that need to be solved and the inverses required in their construction vary widely, and thus the computational effort required depends upon the size of the pedigree, the number of genotyped animals and the number of loci.

Theory: We present computational strategies to avoid storing large, dense blocks of the MME that involve imputed genotypes. Furthermore, we present a hybrid model that fits a MEM for animals with observed genotypes and a BVM for those without genotypes. The hybrid model is computationally attractive for pedigree files containing millions of animals with a large proportion of those being genotyped.

Application: We demonstrate the practicality on both the original MEM and the hybrid model using real data with 6,179,960 animals in the pedigree with 4,934,101 phenotypes and 31,453 animals genotyped at 40,214 informative loci. To complete a single-trait analysis on a desk-top computer with four graphics cards required about 3 h using the hybrid model to obtain both preconditioned conjugate gradient solutions and 42,000 Markov chain Monte-Carlo (MCMC) samples of breeding values, which allowed making inferences from posterior means, variances and covariances. The MCMC sampling required one quarter of the effort when the hybrid model was used compared to the published MEM.

Conclusions: We present a hybrid model that fits a MEM for animals with genotypes and a BVM for those without genotypes. Its practicality and considerable reduction in computing effort was demonstrated. This model can readily be extended to accommodate multiple traits, multiple breeds, maternal effects, and additional random effects such as polygenic residual effects.

Background

Two types of equivalent mixed linear models are used for whole-genome analyses in livestock [1]. The first type, which we refer to as marker effects models (MEM), includes random effects ($\boldsymbol{\alpha}$) of marker genotype covariates ($\mathbf{M}_g$) in the model [2, 3]. The second type, which we refer to as breeding value models (BVM), includes the breeding values of the animals, $\mathbf{u}_g = \mathbf{M}_g\boldsymbol{\alpha}$, as a random effect that has a covariance computed from $\mathbf{M}_g$ [1, 2, 4–6] rather than from the pedigree.

It was shown that the BVM can be adapted for what is known as single-step genomic best linear unbiased

*Correspondence: rohan@iastate.edu
[1] Department of Animal Science, Iowa State University, Ames, IA 50011, USA
Full list of author information is available at the end of the article

prediction (SS-GBLUP) that combines information from animals with genotypes and from those without genotypes in a single BLUP analysis [7–9]. However, the SS-GBLUP analysis requires computing the inverse of **G**, which is the matrix of genomic relationships of the animals with genotypes [8, 9]. When the number N_g of genotyped animals exceeds the number of markers, **G** is singular, but a full-rank matrix such as $\mathbf{G}^* = 0.95\mathbf{G} + 0.05\mathbf{A}$, with **A** being the pedigree-based relationship matrix might be used in its place. Single-step analyses based on the MEM do not require computing **G** or its inverse [10]. Furthermore, Bayesian regression analyses based on the MEM are not limited to assuming a normal prior for $\boldsymbol{\alpha}$, which is implicit in SS-GBLUP; Bayesian regression models can accommodate various priors including the t distribution as in BayesA [3, 11], the double exponential distribution as in Bayesian LASSO [12] or mixtures of the t distribution or the normal distribution [3, 11, 13] as in BayesB or BayesC. However, the MME that correspond to single-step MEM (SS-MEM) types of models contain dense blocks that correspond to the imputed genotypes of animals with missing genotypes [10], and those blocks can be large if many animals have missing genotypes.

Liu et al. [14] developed a single-step method based on the BVM with direct estimation of marker effects (SSME-GBLUP). An advantage of that method over SS-GBLUP is that it does not require computing **G** or its inverse. Also, their method can be used for Bayesian regression models [14]. However, the MME for SSME-GBLUP contains expressions that involve the inverse of the pedigree-based relationship matrix, $\mathbf{A}_{gg}$, for the animals with genotypes. This is a dense matrix, and therefore a computational strategy was proposed to avoid computing its inverse but it requires solving a dense system of equations of order N_g within each round of Jacobi or pre-conditioned conjugate gradient (PCG) iteration for solution of the MME or within each round of MCMC sampling for Bayesian inference with models such as BayesA or BayesB [3]. Equation (A1) in Legarra and Ducrocq [15] also present a set of similar MME with marker effects for genotyped animals and breeding values for non-genotyped animals. As with the MME in Liu et al. [14], the advantage of the MME of Legarra and Ducrocq [15] is that they do not require the computation of **G** or its inverse but require computing the inverse of $\mathbf{A}_{gg}$. Recently, in some livestock such as dairy cattle, N_g has increased towards a million or more, and thus, solving a dense system of equations of order N_g within each round of iteration will place a heavy burden on SSME-GBLUP in computing time and storage requirements.

The objective of this paper is to present computational strategies for whole-genome analyses based on the SS-MEM that avoid storing large, dense blocks of the MME that involve imputed genotypes. First, we will show this for the MME given in [10]. Second, we will present what we refer to as a hybrid type model (HM) that uses a MEM for the animals with marker genotypes and a BVM for animals without genotypes. The MME that correspond to this model also has dense blocks that correspond to animals with missing genotypes. However, in Bayesian regression analyses based on this hybrid model, storing the dense blocks can be avoided even more efficiently than was the case for the MME given in [10]. Finally, we will present the computer storage and time required for a real application.

Theory

In most genomic analyses, the columns of the matrix $\mathbf{M}_g$ of marker covariates are centered to have zero expectations. This ensures that the vector of breeding values, $\mathbf{u}_g = \mathbf{M}_g\boldsymbol{\alpha}$, has a mean of **0**. Centering $\mathbf{M}_g$ requires knowing the expected value of the marker covariates for founder animals. Often, these expected values are unknown, but can be incorporated into the model as a location parameter [10, 16]. However, to simplify our presentation without loss of generality we assume that $\mathbf{M}_g$ is a matrix of correctly centered marker covariates.

Marker effects model for single-step Bayesian regression

As in Fernando et al. [10], a MEM for single-step Bayesian regression analyses can be derived from writing the model equation as:

$$\begin{bmatrix}\mathbf{y}_n\\ \mathbf{y}_g\end{bmatrix} = \begin{bmatrix}\mathbf{X}_n\\ \mathbf{X}_g\end{bmatrix}\boldsymbol{\beta} + \begin{bmatrix}\mathbf{Z}_n & \mathbf{0}\\ \mathbf{0} & \mathbf{Z}_g\end{bmatrix}\begin{bmatrix}\mathbf{M}_n\boldsymbol{\alpha} + \boldsymbol{\epsilon}\\ \mathbf{M}_g\boldsymbol{\alpha}\end{bmatrix} + \mathbf{e}, \quad (1)$$

where the vectors and matrices for animals without genotypes are denoted with a subscript n and those for the animals with genotypes with a subscript g. Thus, $\mathbf{y}_n$ and $\mathbf{y}_g$ are the vectors of phenotypic values, $\mathbf{X}_n$ and $\mathbf{X}_g$ are the incidence matrices for the fixed effects, $\boldsymbol{\beta}$, $\mathbf{Z}_n$ and $\mathbf{Z}_g$ are incidence matrices that relate the breeding values of animals, $\begin{bmatrix}\mathbf{M}_n\boldsymbol{\alpha} + \boldsymbol{\epsilon}\\ \mathbf{M}_g\boldsymbol{\alpha}\end{bmatrix}$, to the phenotypic values, $\mathbf{M}_g$ is the matrix of centered marker covariates for animals with genotypes, $\mathbf{M}_n = \mathbf{A}_{ng}\mathbf{A}_{gg}^{-1}\mathbf{M}_g$, is the matrix of imputed marker covariates for animals with missing genotypes, $\boldsymbol{\alpha}$ is the vector of random marker effects, $\boldsymbol{\epsilon}$ is the vector of imputation residuals with null means and covariance matrix proportional to the inverse of $\mathbf{A}^{nn}$, the sub-matrix corresponding to animals with missing genotypes in the inverse of the matrix **A** of pedigree-based additive relationships, and **e** is a vector of residuals. The matrix of imputed genotypes can be more efficiently computed by solving the sparse system of equations [10]:

$$\mathbf{A}^{nn}\mathbf{M}_n = -\mathbf{A}^{ng}\mathbf{M}_g. \tag{2}$$

Depending on the prior used for $\boldsymbol{\alpha}$, Model (1) can be used for a range of single-step Bayesian regression analyses, including single-step BLUP, BayesA, BayesB, BayesC or Bayesian LASSO [10]. Those models (1) and their corresponding analyses assume that the breeding values can be adequately explained by the marker covariates. If that assumption does not hold, a polygenic residual with a mean of zero and a covariance matrix that is proportional to $\mathbf{A}$ can be included as an additional effect in the model.

The MME that correspond to Model (1) for BayesC with $\pi = 0$ are:

$$\begin{bmatrix} \mathbf{X}'\mathbf{X} & \mathbf{X}'\mathbf{ZM} & \mathbf{X}_n'\mathbf{Z}_n \\ \mathbf{M}'\mathbf{Z}'\mathbf{X} & \mathbf{M}'\mathbf{Z}'\mathbf{ZM} + \mathbf{I}\frac{\sigma_e^2}{\sigma_\alpha^2} & \mathbf{M}_n'\mathbf{Z}_n'\mathbf{Z}_n \\ \mathbf{Z}_n'\mathbf{X}_n & \mathbf{Z}_n'\mathbf{Z}_n\mathbf{M}_n & \mathbf{Z}_n'\mathbf{Z}_n + \mathbf{A}^{nn}\frac{\sigma_e^2}{\sigma_g^2} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{\beta}} \\ \hat{\boldsymbol{\alpha}} \\ \hat{\boldsymbol{\epsilon}} \end{bmatrix} = \begin{bmatrix} \mathbf{X}'\mathbf{y} \\ \mathbf{M}'\mathbf{Z}'\mathbf{y} \\ \mathbf{Z}_n'\mathbf{y}_n \end{bmatrix}, \tag{3}$$

where $\mathbf{X} = \begin{bmatrix} \mathbf{X}_n \\ \mathbf{X}_g \end{bmatrix}$, $\mathbf{Z} = \begin{bmatrix} \mathbf{Z}_n & \mathbf{0} \\ \mathbf{0} & \mathbf{Z}_g \end{bmatrix}$, $\mathbf{M} = \begin{bmatrix} \mathbf{M}_n \\ \mathbf{M}_g \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} \mathbf{y}_n \\ \mathbf{y}_g \end{bmatrix}$, σ_α^2 is the variance of marker effects, σ_g^2 is the additive genetic variance, and σ_e^2 is the residual variance. These Eq. (3) contain matrix-by-matrix products and matrix-by-vector products involving the dense matrix $\mathbf{M}_n$ of imputed genotypes. We will assume here that $\mathbf{X}'\mathbf{ZM}, \mathbf{M}'\mathbf{Z}'\mathbf{X}$ and $\mathbf{M}'\mathbf{Z}'\mathbf{ZM}$ are small enough to be stored in memory. Below, we present computing strategies for calculations that involve $\mathbf{Z}_n'\mathbf{Z}_n\mathbf{M}_n$ or its transpose without storing these large matrices in memory. If the matrices, $\mathbf{X}'\mathbf{ZM}$ and $\mathbf{M}'\mathbf{Z}'\mathbf{X}$, are large, the computing strategies presented below can also be adapted for calculations that involve these matrices as will be done in our example application.

Computing strategies

First, we will discuss the calculations necessary to apply PCG to (3). Following this, we will discuss how to use (3) to obtain Markov chain Monte-Carlo (MCMC) samples of the location parameters of Model (1) from their full conditional distributions.

Preconditioned conjugate gradient iteration The PCG algorithm is widely used to iteratively solve the MME, e.g., [17, 18]. In each iteration of PCG, the left-hand-side of the MME (LHS-MME) is post-multiplied by a vector. However, the LHS-MME given in (3) contains two dense sub-matrices, $\mathbf{Z}_n'\mathbf{Z}_n\mathbf{M}_n$ and its transpose, that may be too large for storage in memory; the remaining sub-matrices in LHS-MME can be stored in memory either because they are not too large or because they are sparse. In each round of PCG, $\mathbf{Z}_n'\mathbf{Z}_n\mathbf{M}_n$ needs to be post-multiplied by a vector $\mathbf{q}$ that has the same order as $\boldsymbol{\alpha}$ and the transpose of this matrix by a vector $\mathbf{s}$ that has the same order as $\boldsymbol{\epsilon}$. The first of these products can be done without storing $\mathbf{Z}_n'\mathbf{Z}_n\mathbf{M}_n$ in memory as follows. Post-multiplying both sides of Eq. (2) by $\mathbf{q}$ gives:

$$\begin{aligned} \mathbf{A}^{nn}\mathbf{M}_n\mathbf{q} &= -\mathbf{A}^{ng}\mathbf{M}_g\mathbf{q} \\ \mathbf{A}^{nn}\mathbf{x} &= \mathbf{b}, \end{aligned} \tag{4}$$

where $\mathbf{x} = \mathbf{M}_n\mathbf{q}$ and $\mathbf{b} = -\mathbf{A}^{ng}(\mathbf{M}_g\mathbf{q})$. Note that for efficient computation, the matrix $\mathbf{M}_g$ is first multiplied by $\mathbf{q}$ and the resulting vector is then premultiplied by the sparse matrix $-\mathbf{A}^{ng}$ to get $\mathbf{b}$. Solving the sparse system (4) gives the product $\mathbf{x} = \mathbf{M}_n\mathbf{q}$ without storing the large dense matrix $\mathbf{M}_n$ in memory, and premultiplying $\mathbf{x}$ by $\mathbf{Z}_n'\mathbf{Z}_n$ gives the first product that is required for PCG. To obtain the second product, note that from Eq. (2),

$$\mathbf{M}_n' = -\mathbf{M}_g'\mathbf{A}^{gn}(\mathbf{A}^{nn})^{-1}. \tag{5}$$

Thus, the required product $\mathbf{M}_n'\mathbf{Z}_n'\mathbf{Z}_n\mathbf{s}$ can be written as $-\mathbf{M}_g'\mathbf{A}^{gn}(\mathbf{A}^{nn})^{-1}\mathbf{Z}_n'\mathbf{Z}_n\mathbf{s}$. To compute this efficiently, first the product $\mathbf{b} = \mathbf{Z}_n'\mathbf{Z}_n\mathbf{s}$ is obtained. Then, solving the sparse system:

$$\mathbf{A}^{nn}\mathbf{x} = \mathbf{b}, \tag{6}$$

gives $\mathbf{x} = (\mathbf{A}^{nn})^{-1}\mathbf{Z}_n'\mathbf{Z}_n\mathbf{s}$, where $\mathbf{b}$ and $\mathbf{x}$ have been reused to denote intermediate results in these computations. Next, $\mathbf{x}$ is premultiplied by the sparse matrix $-\mathbf{A}^{gn}$ and the resulting vector is premultiplied by $\mathbf{M}_g'$ to get the second product that is required for PCG. The remaining matrix-by-vector products for PCG can be obtained directly because these matrices are stored in memory.

We will now describe how these matrices and the right hand sides involving $\mathbf{M}_n$ can be computed in order to form the other elements of the MME without storing $\mathbf{M}_n$ in memory.

Consider computing:

$$\mathbf{M}'\mathbf{Z}'\mathbf{ZM} = \mathbf{M}_n'\mathbf{Z}_n'\mathbf{Z}_n\mathbf{M}_n + \mathbf{M}_g'\mathbf{Z}_g'\mathbf{Z}_g\mathbf{M}_g,$$

without storing $\mathbf{M}_n$ in memory. Let $\mathbf{m}_{n_i}'$ denote row i of $\mathbf{Z}_n\mathbf{M}_n$. Then, $\mathbf{M}_n'\mathbf{Z}_n'\mathbf{Z}_n\mathbf{M}_n$ can be written as:

$$\mathbf{M}_n'\mathbf{Z}_n'\mathbf{Z}_n\mathbf{M}_n = \sum_i \mathbf{m}_{n_i}\mathbf{m}_{n_i}'. \tag{7}$$

Thus, the matrix product $\mathbf{M}_n'\mathbf{Z}_n'\mathbf{Z}_n\mathbf{M}_n$ can be computed without storing $\mathbf{M}_n$ in memory if each row of $\mathbf{Z}_n\mathbf{M}_n$ can be obtained without computing the entire matrix. Rearranging (2), row i of $\mathbf{Z}_n\mathbf{M}_n$ can be computed as:

$$\mathbf{m}_{n_i}' = -\mathbf{e}_i'\mathbf{Z}_n(\mathbf{A}^{nn})^{-1}\mathbf{A}^{ng}\mathbf{M}_g, \tag{8}$$

where $\mathbf{e}'_i$ is a row vector with 1 in the ith position and 0s elsewhere, and the product $\mathbf{e}'_i\mathbf{Z}_n(\mathbf{A}^{nn})^{-1}$ can be obtained by solving the sparse system:

$$\mathbf{A}^{nn}\mathbf{x} = \mathbf{b}, \tag{9}$$

where $\mathbf{b} = \mathbf{Z}'_n\mathbf{e}_i$. Note that the solution to (9) gives $\mathbf{x}' = \mathbf{e}'_i\mathbf{Z}_n(\mathbf{A}^{nn})^{-1}$, without having to invert $\mathbf{A}^{nn}$. These row vectors of $\mathbf{Z}_n\mathbf{M}_n$ can also be used to compute $\mathbf{X}'_n\mathbf{Z}_n\mathbf{M}_n$ as:

$$\mathbf{X}'_n\mathbf{Z}_n\mathbf{M}_n = \sum_i \mathbf{x}_i\mathbf{m}'_{n_i}, \tag{10}$$

where $\mathbf{x}_i$ is used here to denote the ith column of $\mathbf{X}'_n$, which is the first term of

$$\mathbf{X}'\mathbf{Z}\mathbf{M} = \mathbf{X}'_n\mathbf{Z}_n\mathbf{M}_n + \mathbf{X}'_g\mathbf{Z}_g\mathbf{M}_g,$$

which is a product of $\mathbf{M}_n$. Similarly, the right-hand-side vector $\mathbf{M}'\mathbf{Z}'\mathbf{y}$ can be written as the sum:

$$\mathbf{M}'\mathbf{Z}'\mathbf{y} = \mathbf{M}'_n\mathbf{Z}'_n\mathbf{y}_n + \mathbf{M}'_g\mathbf{Z}'_g\mathbf{y}_g,$$

which is a product of $\mathbf{M}_n$, and its first term can be computed as:

$$\mathbf{M}'_n\mathbf{Z}'_n\mathbf{y}_n = \sum_i \mathbf{m}_{n_i}y_i, \tag{11}$$

where y_i is used to denote the ith element of $\mathbf{y}_n$.

Note that computing $\mathbf{m}'_{n_i}$ corresponding to row i of $\mathbf{Z}_n\mathbf{M}_n$ using Eq. (8) can be done independently of its computation for any other row, and thus, the computations in Eqs. (7), (10), and (11) can be easily parallelized.

There are a number of approaches to compute $\mathbf{m}'_{n_i}$ that can be used. One approach for solving Eq. (9) for n less than approximately ten million using a typical workstation computer is to obtain a sparse Cholesky factor of $\mathbf{A}^{nn}$ and directly solve for each $\mathbf{m}'_{n_i}$ using forward and backward substitution. Software libraries exist for obtaining a Cholesky factor of large sparse matrices [19] using multiple threads or general purpose graphics processing units (GPU). Once the factor is obtained, independent threads can be used to solve in parallel from a single memory copy of the factor. Note that the Cholesky factor of $\mathbf{A}^{nn}$ may be denser than $\mathbf{A}^{nn}$, depending upon the nature of the relationships between genotyped and non-genotyped animals. For example, if non-genotyped animals comprise only non-parents, then $\mathbf{A}^{nn}$ is diagonal and solution for $\mathbf{m}'_i$ is trivial. For larger linear systems with non-genotyped parents where the Cholesky factor is too large, indirect solution using high performance methods on GPU is a practical alternative. The PCG algorithm parallelizes well and performs efficiently on GPU.

MCMC sampling Gibbs sampling is a widely used MCMC method for inference with Bayesian regression models, e.g., [3, 20, 21]. One of the most time-consuming tasks in these analyses is single-site sampling of the location parameters from their full-conditional distributions. Let $\boldsymbol{\theta} = \begin{bmatrix}\boldsymbol{\beta}\\ \boldsymbol{\alpha}\\ \boldsymbol{\epsilon}\end{bmatrix}$ denote the location parameters in Model (1). Then, following [22], the full-conditional distribution for θ_i under BayesC with $\pi = 0$ is:

$$\theta_i|\text{ELSE} \sim \text{N}\left(\tilde{\theta}_i, c_{ii}^{-1}\sigma_e^2\right),$$

where ELSE is used to denote all the other parameters in the model and the vector of phenotypes, $\tilde{\theta}_i$ is the solution to:

$$c_{ii}\tilde{\theta}_i = r_i - \mathbf{c}'_i\boldsymbol{\theta} + c_{ii}\theta_i, \tag{12}$$

c_{ii} is the ith diagonal of the matrix $\mathbf{C}$ that denotes the LHS-MME given in (3), r_i is the right-hand-side element from (3) corresponding to θ_i, and $\mathbf{c}'_i$ is row i of $\mathbf{C}$. However, as mentioned previously, some sub-matrices of the LHS-MME given in (3) are dense and too large to be stored in memory. However, as explained below, the same strategy used to avoid storing these sub-matrices in PCG calculations can also be used here.

Consider computing the full conditional mean and variance for $\theta_i = \alpha_j$. Then c_{ii}, the ith diagonal element from $\mathbf{C}$ is obtained from the jth diagonal of $\mathbf{B} = \mathbf{M}'\mathbf{Z}'\mathbf{Z}\mathbf{M} + \mathbf{I}\frac{\sigma_{\mathbf{e}}^2}{\sigma_\alpha^2}$, which can be stored in memory. Similarly, $r_i - \mathbf{c}'_i\boldsymbol{\theta} + c_{ii}\theta_i$ is computed as:

$$r_i - \mathbf{c}'_i\boldsymbol{\theta} + c_{ii}\theta_i = d_j - \mathbf{b}'_j\boldsymbol{\alpha} + b_{jj}\alpha_j,$$

where d_j is element j of the vector

$$\mathbf{d} = \mathbf{M}'\mathbf{Z}'\mathbf{y} - \mathbf{M}'\mathbf{Z}'\mathbf{X}\boldsymbol{\beta} - \mathbf{M}'_n\mathbf{Z}'_n\mathbf{Z}_n\boldsymbol{\epsilon},$$

$\mathbf{b}'_j$ is row j and b_{jj} is the jth diagonal of $\mathbf{B}$. We have already seen how the large, dense matrix $\mathbf{M}'_n\mathbf{Z}'_n\mathbf{Z}_n$ can be multiplied by a vector such as $\boldsymbol{\epsilon}$ without storing this matrix in memory, and this same strategy can be used here to compute $\mathbf{M}'_n\mathbf{Z}'_n\mathbf{Z}_n\boldsymbol{\epsilon}$. The full-conditional distribution for α_j, under BayesC with $\pi = 0$, becomes:

$$\alpha_j|\text{ELSE} \sim \text{N}\left(\tilde{\alpha}_j, b_{jj}^{-1}\sigma_e^2\right),$$

where $\tilde{\alpha}_j$ is the solution to:

$$b_{jj}\tilde{\alpha}_j = d_j - \mathbf{b}'_j\boldsymbol{\alpha} + b_{jj}\alpha_j. \tag{13}$$

The right-hand-side of this Eq. (13) is also used for calculations that involve variable selection in BayesB and BayesC when $\pi > 0$ [21, 23].

Similarly, to compute full-conditional mean and variance for $\theta_i = \epsilon_j$, c_{ii} is obtained from the jth diagonal of $\mathbf{B}$ that now denotes $\mathbf{B} = \mathbf{Z}'_n\mathbf{Z}_n + \mathbf{A}^{nn}\frac{\sigma_e^2}{\sigma_g^2}$, and

$$r_i - \mathbf{c}_i'\boldsymbol{\theta} + c_{ii}\theta_i = d_j - \mathbf{b}'_j\boldsymbol{\epsilon} + b_{jj}\epsilon_j,$$

where d_j is element j of the vector that now denotes:

$$\mathbf{d} = \mathbf{Z}_n'\mathbf{y}_n - \mathbf{Z}_n'\mathbf{X}_n\boldsymbol{\beta} - \mathbf{Z}_n'\mathbf{Z}_n\mathbf{M}_n\boldsymbol{\alpha}.$$

The product $\mathbf{Z}_n'\mathbf{Z}_n\mathbf{M}_n\boldsymbol{\alpha}$ is obtained as described for PCG calculations. Then, the full-conditional distribution for ϵ_j becomes:

$$\epsilon_j|\text{ELSE} \sim \text{N}\left(\tilde{\epsilon}_j, b_{jj}^{-1}\sigma_e^2\right),$$

where $\tilde{\epsilon}_j$ is the solution to

$$b_{jj}\tilde{\epsilon}_j = d_j - \mathbf{b}'_j\boldsymbol{\epsilon} + b_{jj}\epsilon_j.$$

Samples of model effects such as $\boldsymbol{\beta}, \boldsymbol{\alpha}$, and $\boldsymbol{\epsilon}$, or their linear functions that represent breeding values namely $\mathbf{M}_g\boldsymbol{\alpha}$ for genotyped animals or $\mathbf{M}_n\boldsymbol{\alpha} + \boldsymbol{\epsilon}$ for non-genotyped animals can be accumulated as sums and sums of squares to obtain posterior means and prediction error variances. Alternatively, samples of fitted model effects can be written to a file for post-processing.

Hybrid model for single-step Bayesian regression

The large, dense matrix $\mathbf{M}_n$ appears in the MEM given by Model (1). This is avoided here by using a BVM for animals with missing genotypes rather than expressing their breeding values as the sum of the effects of their imputed marker genotypes plus their separate imputation residuals. The advantages of the MEM such as allowing for alternative priors for marker effects are retained by still fitting a MEM but only for animals with genotypes. The hybrid model equation is:

$$\begin{bmatrix} \mathbf{y}_n \\ \mathbf{y}_g \end{bmatrix} = \begin{bmatrix} \mathbf{X}_n \\ \mathbf{X}_g \end{bmatrix}\boldsymbol{\beta} + \begin{bmatrix} \mathbf{0} & \mathbf{Z}_n \\ \mathbf{Z}_g\mathbf{M}_g & \mathbf{0} \end{bmatrix}\begin{bmatrix} \boldsymbol{\alpha} \\ \mathbf{u}_n \end{bmatrix} + \mathbf{e}, \qquad (14)$$

with $\mathbf{u}_n = \mathbf{M}_n\boldsymbol{\alpha} + \boldsymbol{\epsilon}$, and thus, this single-step hybrid model (SS-HM) is equivalent to the SS-MEM (1). To construct the MME for this model (14), we need to invert the covariance matrix corresponding to the random effects, namely $\boldsymbol{\Sigma} = \text{Var}\left(\begin{bmatrix} \boldsymbol{\alpha} \\ \mathbf{u}_n \end{bmatrix}\right)$. That inverse can be obtained by first writing the random effects of (14) as:

$$\begin{bmatrix} \boldsymbol{\alpha} \\ \mathbf{u}_n \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{M}_n & \mathbf{I} \end{bmatrix}\begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\epsilon} \end{bmatrix}.$$

Then, $\boldsymbol{\Sigma}$ can be written as:

$$\begin{aligned}\boldsymbol{\Sigma} &= \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{M}_n & \mathbf{I} \end{bmatrix}\text{Var}\left(\begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\epsilon} \end{bmatrix}\right)\begin{bmatrix} \mathbf{I} & \mathbf{M}_n' \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{M_n} & \mathbf{I} \end{bmatrix}\begin{bmatrix} \mathbf{I}\sigma_\alpha^2 & \mathbf{0} \\ \mathbf{0} & (\mathbf{A}^{nn})^{-1}\sigma_g^2 \end{bmatrix}\begin{bmatrix} \mathbf{I} & \mathbf{M}_n' \\ \mathbf{0} & \mathbf{I} \end{bmatrix},\end{aligned}$$

and its inverse can be obtained as:

$$\begin{aligned}\boldsymbol{\Sigma}^{-1} &= \begin{bmatrix} \mathbf{I} & -\mathbf{M}_n' \\ \mathbf{0} & \mathbf{I} \end{bmatrix}\begin{bmatrix} \mathbf{I}\frac{1}{\sigma_\alpha^2} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}^{nn}\frac{1}{\sigma_g^2} \end{bmatrix}\begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{M}_n & \mathbf{I} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}\frac{1}{\sigma_\alpha^2} + \begin{bmatrix} -\mathbf{M}_n' \\ \mathbf{I} \end{bmatrix}\mathbf{A}^{nn}\frac{1}{\sigma_g^2}\begin{bmatrix} -\mathbf{M}_n & \mathbf{I} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}\frac{1}{\sigma_\alpha^2} + \begin{bmatrix} \mathbf{M}_n'\mathbf{A}^{nn}\mathbf{M}_n & -\mathbf{M}_n'\mathbf{A}^{nn} \\ -\mathbf{A}^{nn}\mathbf{M}_n & \mathbf{A}^{nn} \end{bmatrix}\frac{1}{\sigma_g^2}.\end{aligned}$$

Now, using the result $\mathbf{M}_n = \mathbf{A}_{ng}\mathbf{A}_{gg}^{-1}\mathbf{M}_g = -(\mathbf{A}^{nn})^{-1}\mathbf{A}^{ng}\mathbf{M}_g$ [10], in the off-diagonal blocks of the second term, $\boldsymbol{\Sigma}^{-1}$ becomes:

$$\boldsymbol{\Sigma}^{-1} = \begin{bmatrix} \mathbf{I}\frac{1}{\sigma_\alpha^2} + \mathbf{M}_n'\mathbf{A}^{nn}\mathbf{M}_n\frac{1}{\sigma_g^2} & \mathbf{M}_g'\mathbf{A}^{gn}\frac{1}{\sigma_g^2} \\ \mathbf{A}^{ng}\mathbf{M}_g\frac{1}{\sigma_g^2} & \mathbf{A}^{nn}\frac{1}{\sigma_g^2} \end{bmatrix},$$

and then the MME for the HM (14) can be written as:

$$\begin{bmatrix} \mathbf{X}'\mathbf{X} & \mathbf{X}_g'\mathbf{Z}_g\mathbf{M}_g & \mathbf{X}_n'\mathbf{Z}_n \\ \mathbf{M}_g'\mathbf{Z}_g'\mathbf{X}_g & \mathbf{Q} & \mathbf{M}_g'\mathbf{A}^{gn}\frac{\sigma_e^2}{\sigma_g^2} \\ \mathbf{Z}_n'\mathbf{X}_n & \mathbf{A}^{ng}\mathbf{M}_g\frac{\sigma_e^2}{\sigma_g^2} & \mathbf{Z}_n'\mathbf{Z}_n + \mathbf{A}^{nn}\frac{\sigma_e^2}{\sigma_g^2} \end{bmatrix}\begin{bmatrix} \hat{\boldsymbol{\beta}} \\ \hat{\boldsymbol{\alpha}} \\ \hat{\mathbf{u}}_n \end{bmatrix} = \begin{bmatrix} \mathbf{X}'\mathbf{y} \\ \mathbf{M}_g'\mathbf{Z}_g'\mathbf{y}_g \\ \mathbf{Z}_n'\mathbf{y}_n \end{bmatrix}, \qquad (15)$$

where $\mathbf{Q} = \mathbf{M}_g'\mathbf{Z}_g'\mathbf{Z}_g\mathbf{M}_g + \mathbf{I}\frac{\sigma_e^2}{\sigma_\alpha^2} + \mathbf{M}_n'\mathbf{A}^{nn}\mathbf{M}_n\frac{\sigma_e^2}{\sigma_g^2}$. These equations involve $\mathbf{M}_g$ rather than $\mathbf{M}_n$, except in $\mathbf{Q}$, the diagonal block that corresponds to $\hat{\boldsymbol{\alpha}}$, which has dimension equal to the number of marker covariates, often less than 50,000, regardless of the number of genotyped or non-genotyped animals. Furthermore, we assume here that $\mathbf{X}_g'\mathbf{Z}_g\mathbf{M}_g$ and $\mathbf{M}_g'\mathbf{Z}_g'\mathbf{X}_g$ are small enough to be stored in memory.

The only difference between Eq. (15) and the MME given by Equation (A1) in Legarra and Ducrocq [15] is in $\mathbf{Q}$. Using the notation in this paper, the matrix expression $\mathbf{M}_n'\mathbf{A}^{nn}\mathbf{M}_n$ that is present in the $\mathbf{Q}$ is expressed as $\mathbf{M}_g'(\mathbf{A}^{gg} - \mathbf{A}_{gg}^{-1})\mathbf{M}_g$ in that paper [15], which involves the inverse of $\mathbf{A}_{gg}$ that is difficult to compute. However, these two expressions are identical because $(\mathbf{A}^{gg} - \mathbf{A}_{gg}^{-1}) = \mathbf{A}^{gn}(\mathbf{A}^{nn})^{-1}\mathbf{A}^{ng}$ and $\mathbf{M}_n = -(\mathbf{A}^{nn})^{-1}\mathbf{A}^{ng}\mathbf{M}_g$.

Computing strategies

The matrix $\mathbf{M}_n$ of imputed genotypes does not appear alone in the MME (15), but the MME involve rather the matrix product $\mathbf{M}_n'\mathbf{A}^{nn}\mathbf{M}_n$. However, $\mathbf{M}_n'\mathbf{A}^{nn}\mathbf{M}_n$ can be computed efficiently without needing to store the entire $\mathbf{M}_n$ matrix in memory, in situations when the number of genotyped animals is less than the number of non-genotyped animals. To do so, first from Eq. (5) $\mathbf{M}_n'\mathbf{A}^{nn} = -\mathbf{M}_g'\mathbf{A}^{gn}$. Next, column i of $\mathbf{M}_n$ is obtained by solving the sparse system (2) for column i and premultiply it by the sparse matrix $\mathbf{A}^{gn}$. This gives column i of the product $\mathbf{A}^{gn}\mathbf{M}_n$, which has the same size as $\mathbf{M}_g$. The columns of $\mathbf{A}^{gn}\mathbf{M}_n$ can be computed one at a time or in parallel. Premultiplying $\mathbf{A}^{gn}\mathbf{M}_n$ by $-\mathbf{M}'_g$ gives:

$$-\mathbf{M}'_g\mathbf{A}^{gn}\mathbf{M}_n = \mathbf{M}'_n\mathbf{A}^{nn}\mathbf{M}_n. \tag{16}$$

This needs to be done only once to set up the MME and has order equal to the number of marker genotypes which is often much less than the number of genotyped or non-genotyped animals.

These MME also contain two large, dense sub-matrices, namely $\mathbf{A}^{ng}\mathbf{M}_g$ and its transpose. As described previously, in the PCG iteration and in the Gibbs sampling, these matrices need to be post-multiplied by a vector. When the number of genotyped animals is sufficiently smaller than the number of non-genotyped animals, these matrix-by-vector products can be obtained more efficiently by storing in memory the sparse matrix $\mathbf{A}^{ng}$ and the dense but smaller matrix $\mathbf{M}_g$ rather than their product $\mathbf{A}^{ng}\mathbf{M}_g$. In each round of PCG iteration or Gibbs sampling, the matrix-by-vector product $\mathbf{A}^{ng}\mathbf{M}_g\mathbf{q}$, for example, is obtained by first multiplying the dense matrix $\mathbf{M}_g$ by the vector $\mathbf{q}$ and then premultiplying the result by the sparse matrix $\mathbf{A}^{ng}$. The corresponding calculation for the MEM required solving sparse systems of equations given by Eq. (4) in each round of PCG or Gibbs sampling, in addition to the two matrix-by-vector multiplications that are also required here.

In situations when the number of genotyped animals exceeds the number of non-genotyped animals, using SS-MEM that explicitly involves $\mathbf{M}_n$ in off-diagonal blocks may be competitive with SS-HM.

Application of hybrid model

An example dataset from the American Simmental Association is used to demonstrate the computing effort to obtain PCG samples from (15) and the relative computing effort to obtain MCMC samples for the MME of (15) compared to (3). The vector of phenotypes comprised of 4,934,101 birth weight observations; there were 6,179,960 animals in the pedigree file; 31,453 animals in the pedigree file were genotyped and 23,290 of those had birth weight observations. After filtering marker covariates for low minor allele frequency, 40,214 marker effects were included in the model. There were 399,036 fixed effects, including the herd-year-season effects defined in the same manner as in the routine national evaluation. To keep this presentation that compares the computational effort involved in fitting (3) and (15) simple, our application was limited to a single trait ignoring maternal genetic and permanent environmental effects. Furthermore, we did not include a comparison with SS-GBLUP since that model cannot accommodate mixture priors for marker effects as used in this example.

The analyses were performed using a workstation built on an ASUS X99E WS motherboard, a Xeon E5-1650V3 3.5 Ghz processor overclocked to 4.2 Ghz, 128 GB of DDR4 ECC RAM at 2133 Mhz and four NVidia Titan X GPU, with 9TB of workspace in a RAID5 configuration comprising four SATA disks. The operating system was Ubuntu 14.04 LTS, and the BOLT software package (http://manual.thetasolutionsllc.com/IntroBolt) built with the CUDA Toolkit 7.5 was used.

The vector $\mathbf{y}$, and matrices $\mathbf{X}, \mathbf{Z}, \mathbf{A}^{nn}, \mathbf{A}^{gn}, \mathbf{A}^{ng}$, and $\mathbf{M}_g$ were built from data files using BOLT tools. Ordering the pedigree file, construction of $\mathbf{A}^{-1}$, including calculation of inbreeding, and its partitioning into blocks representing genotyped and non-genotyped animals took 3 min and required 1.0 Gb of disk storage and 302 Mb of memory. While $\mathbf{A}^{-1}$ was being formed, $\mathbf{y}$, $\mathbf{X}$ and $\mathbf{Z}$ were created in about 10 s and required 38, 78 and 83 Mb of disk storage. When stored in memory, they required 19.7, 59.2 and 59.2 Mb respectively. The matrix $\mathbf{M}_g$ required 4.2 Gb of disk and memory when stored in single precision.

The matrix product $\mathbf{X}'_g\mathbf{Z}_g\mathbf{M}_g$ and its transpose $\mathbf{M}'_g\mathbf{Z}'_g\mathbf{X}_g$ were not explicitly formed, instead computations involving those terms were done in parts as described previously in this paper. The sparse Cholesky decomposition of the 6,148,507 order $\mathbf{A}^{nn}$ matrix took just under 4 min. The imputation of $\mathbf{M}_n$, using forward and backward substitution with the Cholesky factor, and its premultiplication by $\mathbf{A}^{gn}$ took just over 35 min using eight parallel processes. The creation of the matrix products $\mathbf{M}'_g\mathbf{Z}'_g\mathbf{Z}_g\mathbf{M}_g$ and $\mathbf{M}'_n\mathbf{A}^{nn}\mathbf{M}_n$ each took about 20 s using 2 GPU after obtaining the imputed values and required 6.2 Gb of disk storage and memory when stored in single precision.

For the analysis using Eq. (15) the PCG solution of the MME stored in double precision took just under 40 min, using a single GPU and diagonal preconditioning. Because the PCG was performed in double precision just under 18 Gb of memory was required to store all the submatrices comprising the left-hand side. The right-hand side required 53Mb of memory. Additional memory for work space of approximately 4 Gb was required for PCG. Convergence was determined by comparing solutions from every 200 rounds of iteration to solutions from 5000 rounds. By 1800 rounds the correlation and regression of solutions with those from 5000 rounds were very close to one (.99 each). The PCG residual value was near 1.1e–05. The PCG solution does not give the posterior mean of the marker effects for a model with mixture priors, but was used to define starting values for MCMC sampling of all the effects in the MME, but using a mixture prior for marker effects.

Starting with the same PCG solution but different random number generator seed values, using 4 parallel chains each drawing 10,500 samples on its own GPU took 70 min to obtain a total of 42,000 Gibbs samples, using $\pi = 0.95$ and known variance ratios in Eqs. (3) and (15). Each of the parallel Gibbs Sampler jobs shared a single copy in shared memory of the left-hand-side matrices, reducing the memory requirements and reading from disk.

Experience has shown that 40,000 samples after burn-in is sufficient to obtain posterior means of breeding values that are stable for the hybrid model and MEM. However, we confirmed this by sampling four additional parallel chains each with length 250,000 samples after a 5000 sample burn. The purpose of these very long chains was to confirm that the 40,000 length post-burn-in chain was sufficient, thus supporting the timings provided here to achieve useful results. The correlation of the posterior means of the breeding values for genotyped animals and non-genotyped animals from the aggregated 40,000 length chain was .99 and 1.0, respectively, with the posterior means from the aggregated 1,000,000 length chain. However, a chain longer that 40,000 may be needed to accurately estimate PEV for animals with intermediate to low accuracies. Because only off-diagonal blocks of the left-hand side are used in the GPU computation for updating the right-hand-sides for each vector of single-site Gibbs samples for $\boldsymbol{\beta}$, $\boldsymbol{\alpha}$ or $\mathbf{u}_n$, and the Gibbs samples were obtained using single precision, the entire left-hand side without the diagonal blocks fit on the GPU. This strategy also allowed the GPU to asynchronously update the right-hand-side while the next set of effects was being sampled using the CPU.

The Gibbs sampler was performed using single precision for storage of the left- and right-hand sides, requiring approximately half as much memory as the 18 Gb required for the PCG which was performed in double precision. Additional memory for work space of approximately 2 Gb was required for the Gibbs sampler. The total time required to assemble the left- and right-hand sides, after the matrix components were formed, was just under 3 min. The total job time for all steps, starting with the raw data, to obtain posterior mean estimates of the MCMC samples of marker effects and MCMC samples of breeding values of the genotyped and non-genotyped animals and their prediction error variances (from the posterior variances of their MCMC samples), took approximately 3 h.

The memory required to store $\mathbf{M}_g$ is determined by the product of the number of animals genotyped and the number of marker covariates. A compressed dense format (CBRC) allows this matrix to be 32 times larger than with the double precision version used above, but increased the computing time for PCG in this example by 25%.

An additional Gibbs sampler run was made with the MME of (3) that used Eq. (4), which requires within each iteration, forward and backward solves using the factor of $\mathbf{A}^{nn}$. The time required to obtain one sample of all effects was 2.0 s. Using the MME of (15) required only 0.44 s for each sample of all effects. Accordingly, the hybrid model has considerable advantage over that of [10]. These two computing approaches should give the same estimates of breeding values as they represent equivalent models as explained in the theory section. The correlations between the MCMC-derived estimates of breeding values between the two approaches were 1.0 for non-genotyped animals and over 0.99 for genotyped animals.

Computational performance of Eq. (15) was compared ignoring the genotypes on approximately half the genotyped animals to demonstrate the effect of the proportion of genotyped animals on computing time. This reduced dataset left 15,694 animals with genotype information of which 11,683 had a birth weight observation. The total number of animals in the pedigree file and number of observations on birth weight were the same as before. After filtering the marker covariates for low minor allele frequency, 40,211 marker loci remained. The time necessary to complete the PCG solver was about 3 min less than the 27 min needed for the larger analysis, which had approximately double the number of genotyped animals. The reduction in time necessary to complete the PCG solver was primarily due to reductions in time used for matrix multiplications involving the smaller matrix $\mathbf{M}_g$. The time necessary to obtain the 42,000 Gibbs samples was reduced by about 20% to 1 h. Imputation required 24 instead of 35 min. Creating $\mathbf{M}'_g\mathbf{Z}'_g\mathbf{Z}_g\mathbf{M}_g$ and $\mathbf{M}'_n\mathbf{A}^{nn}\mathbf{M}_n$ required just under 20 s, the same as before. Thus, doubling the proportion of genotyped animals increased the total job time from about 2.5 to 3 h.

Discussion

Fernando et al. [10] introduced a single-step MEM that is equivalent to SS-GBLUP in the special case when all markers are fitted in the model. It has the advantage compared to SS-GBLUP that it can accommodate a wider class of models with different priors including mixture distributions. However, the MME corresponding to that model includes large, dense off-diagonal sub-matrices, $\mathbf{Z}'_n\mathbf{Z}_n\mathbf{M}_n$ and its transpose, between the blocks for marker effects and the imputation residuals. These sub-matrices are prohibitively large from a storage and computational viewpoint when there is a large number of non-genotyped animals. We have shown here that these limitations can be circumvented by representing those sub-matrices as:

$$-\mathbf{Z}'_n\mathbf{Z}_n(\mathbf{A}^{nn})^{-1}\mathbf{A}^{ng}\mathbf{M}_g$$

and its transpose, and by doing matrix multiplication in parts. This is possible for large problems but requires repeated solutions of an equation of the form $\mathbf{A}^{nn}\mathbf{x} = \mathbf{b}$. A similar solution is used in every iteration of SS-GBLUP when an APY inverse is exploited [18]. Nevertheless, the model in [10] is practical for realistic problems as demonstrated. It does not require approximations [18] as in SS-GBLUP when large numbers of animals are genotyped.

Here we have introduced a single-step HM that is equivalent to the MEM in [10] and in a special case equivalent to SS-GBLUP. The HM includes marker effects and breeding values, and the off-diagonal sub matrices comprise the term $\mathbf{A}^{ng}\mathbf{M}_g$. Computations that involve this sub-matrix can be done efficiently in parts without having to solve equations of the form $\mathbf{A}^{nn}\mathbf{x} = \mathbf{b}$.

The off-diagonal sub-matrices in both these models are the same size, the lower off-diagonal matrix being of the order of the number of non-genotyped animals by the number of markers. In SS-HM, this sub-matrix has a more convenient structure for storage and computation than is generally the case for SS-MEM. We have demonstrated its practicality and its considerable reduction in computing effort. This model can be readily extended to accommodate multiple traits, multiple breeds, maternal effects, and additional random effects such as polygenic residual effects.

Authors' contributions

DJG developed the computational strategies for the MEM and also proposed the HM. RLF, HC and DJG derived the MME for the HM. BLG wrote the software and performed all the data analyses. All authors read and approved the final manuscript.

Author details

[1] Department of Animal Science, Iowa State University, Ames, IA 50011, USA. [2] Theta Solutions, LLC, Atascadero, CA 93422, USA. [3] Institute of Veterinary, Animal and Biomedical Sciences, Massey University, Palmerston North, New Zealand.

Acknowledgements

We are grateful to the reviewers of this manuscript for several helpful suggestions and for pointing out the relationship between the MME for the hybrid model (15) and the MME in [A1] of Legarra and Ducrocq [15]. The authors thank the American Simmental Association for sharing their data. This work was supported in part by the US Department of Agriculture, Agriculture and Food Research Initiative National Institute of Food and Agriculture Competitive Grant No. 2015-67015-22947.

Competing interests

DJG and BLG are partners in Theta Solutions LLC that developed the BOLT software used in this paper. The authors declare that they have no competing interests.

Received: 8 June 2016 Accepted: 22 November 2016
Published online: 08 December 2016

References

1. Strandén I, Garrick DJ. Technical note: Derivation of equivalent computing algorithms for genomic predictions and reliabilities of animal merit. J Dairy Sci. 2009;92:2971–5.
2. Fernando RL. Genetic evaluation and selection using genotypic, phenotypic and pedigree information. In: Proceedings of the 6th World Congress on Genetics Applied to Livestock Production: 11–16 January 1998. vol. 26. Armidale; 1998. pp. 329–36.
3. Meuwissen THE, Hayes BJ, Goddard ME. Prediction of total genetic value using genome-wide dense marker maps. Genetics. 2001;157:1819–29.
4. Nejati-Javaremi A, Smith C, Gibson JP. Effect of total allelic relationship on accuracy of evaluation and response to selection. J Anim Sci. 1997;75:1738–45.
5. Habier D, Fernando RL, Dekkers JCM. The impact of genetic relationship information on genome-assisted breeding values. Genetics. 2007;177:2389–97.
6. VanRaden PM. Efficient methods to compute genomic predictions. J Dairy Sci. 2008;91:4414–23.
7. Legarra A, Aguilar I, Misztal I. A relationship matrix including full pedigree and genomic information. J Dairy Sci. 2009;92:4656–63.
8. Christensen OF, Lund MS. Genomic prediction when some animals are not genotyped. Genet Sel Evol. 2010;42:2.
9. Aguilar I, Misztal I, Johnson DL, Legarra A, Tsuruta S, Lawlor TJ. Hot topic: a unified approach to utilize phenotypic, full pedigree, and genomic information for genetic evaluation of Holstein final score. J Dairy Sci. 2010;93:743–52.
10. Fernando RL, Dekkers JCM, Garrick DJ. A class of Bayesian methods to combine large numbers of genotyped and non-genotyped animals for whole-genome analyses. Genet Sel Evol. 2014;46:59.
11. Gianola D, de los Campos G, Hill WG, Manfredi E, Fernando R. Additive genetic variability and the Bayesian alphabet. Genetics. 2009;83:347–63.
12. de los Campos G, Naya H, Gianola D, Crossa J, Legarra A, Manfredi E, et al. Predicting quantitative traits with regression models for dense molecular markers and pedigree. Genetics. 2009;182:375–85.
13. Habier D, Fernando RL, Kizilkaya K, Garrick DJ. Extension of the Bayesian alphabet for genomic selection. In: Proceedings of the 9th world congress on genetics applied to livestock production: 1–6 August 2010. Leipzig 2010.
14. Liu Z, Goddard ME, Reinhardt F, Reents R. A single-step genomic model with direct estimation of marker effects. J Dairy Sci. 2014;97:5833–50.
15. Legarra A, Ducrocq V. Computational strategies for national integration of phenotypic, genomic, and pedigree data in a single-step best linear unbiased prediction. J Dairy Sci. 2012;95:4629–45.
16. Vitezica ZG, Aguilar I, Misztal I, Legarra A. Bias in genomic predictions for populations under selection. Genet Res (Camb). 2011;93:357–66.
17. Strandén I, Lidauer M. Solving large mixed linear models using preconditioned conjugate gradient iteration. J Dairy Sci. 1999;82:2779–87.
18. Masuda Y, Misztal I, Tsuruta S, Legarra A, Aguilar I, Lourenco DAL, et al. Implementation of genomic recursions in single-step genomic best linear unbiased predictor for US Holsteins with a large number of genotyped animals. J Dairy Sci. 2016;99:1968–74.
19. Chen Y, Davis TA, Hager WW, Rajamanickam S. Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate. ACM Trans Math Softw. 2008;35:22.
20. Habier D, Fernando RL, Kizilkaya K, Garrick DJ. Extension of the Bayesian alphabet for genomic selection. BMC Bioinformatics. 2011;12:186.
21. Fernando R, Garrick D. Bayesian methods applied to GWAS. In: Gondro C, van der Werf J, Hayes B, editors. Genome-wide association studies and genomic prediction. New York: Humana Press; 2013.
22. Sorensen DA, Gianola D. Likelihood, Bayesian, and MCMC methods in quantitative genetics. New York: Springer; 2002.
23. Cheng H, Qu L, Garrick DJ, Fernando RL. A fast and efficient Gibbs sampler for BayesB in whole-genome analyses. Genet Sel Evol. 2015;47:80.