Genet. Sel. Evol. 30 (1998) 3–26

Original article

Restricted maximum likelihood estimation of covariances in sparse linear models

Arnold Neumaier [a], Eildert Groeneveld [b*]

[a] Institut für Mathematik, Universität Wien, Strudlhofgasse 4, 1090 Vienna, Austria
[b] Institut für Tierzucht und Tierverhalten, Bundesforschungsanstalt für Landwirtschaft Höltystr. 10, 31535 Neustadt, Germany

(Received 16 December 1996; accepted 30 September 1997)

Abstract – This paper discusses the restricted maximum likelihood (REML) approach for the estimation of covariance matrices in linear stochastic models, as implemented in the current version of the VCE package for covariance component estimation in large animal breeding models. The main features are: 1) the representation of the equations in an augmented form that simplifies the implementation; 2) the parametrization of the covariance matrices by means of their Cholesky factors, thus automatically ensuring their positive definiteness; 3) explicit formulas for the gradients of the REML function for the case of large and sparse model equations with a large number of unknown covariance components and possibly incomplete data, using the sparse inverse to obtain the gradients cheaply; 4) use of model equations that make separate formation of the inverse of the numerator relationship matrix unnecessary. Many large scale breeding problems were solved with the new implementation, among them an example with more than 250 000 normal equations and 55 covariance components, taking 41 h CPU time on a Hewlett Packard 755. © Inra/Elsevier, Paris

restricted maximum likelihood / variance component estimation / missing data / sparse inverse / analytical gradients

Résumé – Estimation par maximum de vraisemblance restreinte de covariance dans les systèmes linéaires peu denses. Ce papier discute de l'approche par maximum de vraisemblance restreinte (REML) pour l'estimation des matrices de covariances dans les modèles linéaires, qu'applique le logiciel VCE en génétique animale. Les caractéristiques principales sont : 1) la représentation des équations sous forme augmentée qui simplifie les calculs ; 2) le reparamétrage des matrices de variance–covariance grâce aux facteurs de Cholesky qui assure leur caractère défini positif ; 3) les formules explicites des gradients de la fonction REML dans le cas des systèmes d'équations de grande dimension et peu denses avec un grand nombre de composantes de covariances inconnues et éventuellement des données manquantes : elles utilisent les inverses peu denses pour obtenir les gradients de

* Correspondence and reprints

manière économique ; 4) l'utilisation des équations du modèle qui dispense de la formation séparée de l'inverse de la matrice de parenté. Des problèmes de génétique à grande échelle ont été résolus avec la nouvelle version, et parmi eux un exemple avec plus de 250 000 équations normales et 55 composantes de covariance, demandant 41 h de CPU sur un Hewlett Packard 755. © Inra/Elsevier, Paris

maximum de vraisemblance restreinte / estimation des composantes de variance / données manquantes / inverse peu dense / gradient analytique

1. INTRODUCTION

Best linear unbiased prediction of genetic merit [25] requires the covariance structure of the model elements involved. In practical situations, these are usually unknown and must be estimated. During recent years restricted maximum likelihood (REML) [22, 42] has emerged as the method of choice in animal breeding for variance component estimation [15–17, 34–36].

Initially, the expectation maximization (EM) algorithm [6] was used for the optimization of the REML objective function [26, 47].

In 1987 Graser et al. [14] introduced derivative-free optimization, which in the following years led to the development of rather general computing algorithms and packages [15, 28, 29, 34] that were mostly based on the simplex algorithm of Nelder and Mead [40]. Kovac [29] made modifications that turned it into a stable algorithm that no longer converged to noncritical points, but this did not improve its inherent inefficiency for increasing dimensions. Ducos et al. [7] used for the first time the more efficient quasi-Newton procedure approximating gradients by finite differences. While this procedure was faster than the simplex algorithm it was also less robust for higher-dimensional problems because the covariance matrix could become indefinite, often leading to false convergence. Thus, either for lack of robustness and/or excessive computing time often only subsets of the covariance matrices could be estimated simultaneously.

A comparison of different packages [45] confirmed the general observation of Gill [13] that simplex-based optimization algorithms suffer from lack of stability, sometimes converging to noncritical points while breaking down completely at more than three traits. On the other hand the quasi-Newton procedure with optimization on the Cholesky factor as implemented in a general purpose VCE package [18] was stable and much faster than any of the other general purpose algorithms. While this led to a speed-up of between two for small problems and (for some examples) 200 times for larger ones as compared to the simplex procedure, approximating gradients on the basis of finite differences was still exceedingly costly for higher dimensional problems [17].

It is well-known that optimization algorithms generally perform better with analytic gradients if the latter are cheaper to compute than finite difference approximations.

In this paper we derive, in the context of a general statistical model, cheap analytical gradients for problems with a large number p of unknown covariance components using sparse matrix techniques. With hardly any additional storage requirements, the cost of a combined function and gradient evaluation is only three times that of the function value alone. This gives analytic gradients a huge

advantage over finite difference gradients. Misztal and Perez-Enciso [39] investigated the use of sparse matrix technique in the context of an EM algorithm which is known to have much worse convergence properties as compared to quasi-Newton (see also Thompson et al. [48] for an improvement in its space complexity), using an LDL^T factorization and the Takahashi inverse [9]; no results in a REML application were given. A recent papers by Wolfinger et al. [50] (based again on the W transformation) and Meyer [36] (based on the simpler REML objective formulation of Graser et al. [14]) also provide gradients (and even Hessians), but there a gradient computation needs a factor of $O(p)$ more work and space than in our approach, where the complete gradient is found with hardly any additional space and with (depending on the implementation) two to four times the work for a function evaluation.

Meyer [37] used her analytic second derivatives in a Newton-Raphson algorithm for optimization. Because the optimization was not restricted to positive definite covariance matrix approximations (as our algorithm does), she found the algorithm to be markedly less robust than (the already not very robust) simplex algorithm, even for univariate models.

We test the usefulness of our new formulas by integrating them into the VCE covariance component estimation package for animal (and plant) breeding models [17]. Here the gradient routine is combined with a quasi-Newton optimization method and with a parametrization of the covariance parameters by the Cholesky factor that ensures definiteness of the covariance matrix. In the past, this combination was most reliable and had the best convergence properties of all techniques used in this context [45]. Meanwhile, VCE is being used widely in animal and even plant breeding.

In the past, the largest animal breeding problem ever solved ([21], using a quasi-Newton procedure with optimization on the Cholesky factor) comprised 233 796 linear unknowns and 55 covariance components and required 48 days of CPU time on a 100 MHz HP 9000/755 workstation. Clearly, speeding up the algorithm is of paramount importance. In our preliminary implementation of the new method (not yet optimized for speed), we successfully solved this (and an even larger problem of more than 257 000 unknowns) in only 41 h of CPU time, with a speed-up factor of nearly 28 with respect to the finite difference approach.

The new VCE implementation is available free of charge from the ftp site `ftp://192.108.34.1/pub/vce3.2/`. It has been applied successfully throughout the world to hundreds of animal breeding problems, with comparable performance advantages [1–3, 19, 21, 38, 46, 49].

In section 2 we fix notation for linear stochastic models and mixed model equations, define the REML objective function, and review closed formulas for its gradient and Hessian. In sections 3 and 4 we discuss a general setting for practical large scale modeling, and derive an efficient way for the calculation of REML function values and gradients for large and sparse linear stochastic models.

All our results are completely general, not restricted to animal breeding. However, for the formulas used in our implementation, it is assumed that the covariance matrices to be estimated are block diagonal with no restrictions on the (distinct) diagonal blocks.

The final section 5 applies the method to a simple demonstration case and several large animal breeding problems.

2. LINEAR STOCHASTIC MODELS AND RESTRICTED LOGLIKELIHOOD

Many applications (including those to animal breeding) are based on the generalized linear stochastic model

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u} + \boldsymbol{\eta}, \quad \operatorname{cov}(\mathbf{u}) = \mathbf{G}, \quad \operatorname{cov}(\boldsymbol{\eta}) = \mathbf{D}, \tag{1}$$

with fixed effects $\boldsymbol{\beta}$, random effects $\mathbf{u}$ and noise $\boldsymbol{\eta}$. Here $\operatorname{cov}(\mathbf{u})$ denotes the covariance matrix of a random vector $\mathbf{u}$ with zero mean. Usually, $\mathbf{G}$ and $\mathbf{D}$ are block diagonal, with many identical blocks.

By combining the two noise terms, the model is seen to be equivalent to the simple model $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\eta}'$, where $\boldsymbol{\eta}'$ is a random vector with zero mean and (mixed model) covariance matrix $\mathbf{V} = \mathbf{Z}\mathbf{G}\mathbf{Z}^T + \mathbf{D}$. Usually, $\mathbf{V}$ is huge and no longer block diagonal, leading to hardly manageable normal equations involving the inverse of $\mathbf{V}$. However, Henderson [24] showed that the normal equations are equivalent to the mixed model equations

$$\begin{pmatrix} \mathbf{X}^T\mathbf{D}^{-1}\mathbf{X} & \mathbf{X}^T\mathbf{D}^{-1}\mathbf{Z} \\ \mathbf{Z}^T\mathbf{D}^{-1}\mathbf{X} & \mathbf{Z}^T\mathbf{D}^{-1}\mathbf{Z} + \mathbf{G}^{-1} \end{pmatrix} \begin{pmatrix} \boldsymbol{\beta} \\ \mathbf{u} \end{pmatrix} = \begin{pmatrix} \mathbf{X}^T\mathbf{D}^{-1}\mathbf{y} \\ \mathbf{Z}^T\mathbf{D}^{-1}\mathbf{y} \end{pmatrix} \tag{2}$$

This formulation avoids the inverse of the mixed model covariance matrix $\mathbf{V}$ and is the basis of most modern methods for obtaining estimates of $\mathbf{u}$ and $\boldsymbol{\beta}$ in equation (1).

Fellner [10] observed that Henderson's mixed model equations are the normal equations of an augmented model of the simple form

$$\mathbf{A}\mathbf{x} = \mathbf{b} + \boldsymbol{\varepsilon}, \quad \operatorname{cov}(\boldsymbol{\varepsilon}) = \mathbf{C} \tag{3}$$

where

$$\mathbf{x} = \begin{pmatrix} \boldsymbol{\beta} \\ \mathbf{u} \end{pmatrix}, \quad \boldsymbol{\varepsilon} = \begin{pmatrix} -\boldsymbol{\eta} \\ \mathbf{u} \end{pmatrix}, \quad \mathbf{A} = \begin{pmatrix} \mathbf{X} & \mathbf{Z} \\ 0 & \mathbf{I} \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} \mathbf{y} \\ 0 \end{pmatrix}, \quad \text{and} \quad \mathbf{C} = \begin{pmatrix} \mathbf{D} & 0 \\ 0 & \mathbf{G} \end{pmatrix}$$

Thus, without loss in generality, we may base our algorithms on the simple model [3], with a covariance matrix $\mathbf{C}$ that is typically block diagonal. This automatically produces the formulas that previously had to be derived in a less transparent way by means of the W transformation; cf. [5, 11, 23, 50].

The 'normal equations' for the model [3] have the form

$$\mathbf{B}\mathbf{x} = \mathbf{a} \tag{4}$$

where

$$\mathbf{B} = \mathbf{A}^T\mathbf{C}^{-1}\mathbf{A}, \quad \mathbf{a} = \mathbf{A}^T\mathbf{C}^{-1}\mathbf{b}$$

Here $\mathbf{A}^T$ denotes the transposed matrix of $\mathbf{A}$. By solving the normal equations (4), we obtain the best linear unbiased estimate (BLUE) and, for the predictive variables, the best linear unbiased prediction (BLUP)

$$\hat{\mathbf{x}} = \mathbf{B}^{-1}\mathbf{a} = \mathbf{B}^{-1}\mathbf{A}^T\mathbf{C}^{-1}\mathbf{b} \tag{5}$$

for the vector $\mathbf{x}$, and the noise $\boldsymbol{\varepsilon} = \mathbf{A}\mathbf{x} - \mathbf{b}$ is estimated by the residual

$$\mathbf{r} = \mathbf{A}\hat{\mathbf{x}} - \mathbf{b}$$

If the covariance matrix $\mathbf{C} = \mathbf{C}(\omega)$ contains unknown parameters ω (which we shall call 'dispersion parameters', these can be estimated by minimizing the 'restricted loglikelihood'

$$f = \mathbf{r}^T\mathbf{C}^{-1}\mathbf{r} + \log\det\mathbf{C} + \log\det\mathbf{B} \tag{6}$$

quoted in the following as the 'REML objective function', as a function of the parameters ω. (Note that all quantities in the right-hand side of equation (6) depend on $\mathbf{C}$ and hence on ω.)

More precisely, equation (6) is the logarithm of the restricted likelihood, scaled by a factor of $-\frac{1}{2}$ and shifted by a constant depending only on the problem dimension. Under the assumption of Gaussian noise, the restricted likelihood can be derived from the ordinary likelihood restricted to a maximal subspace of independent error contrasts (cf. Harville [22]; our formula (6) is the special case of his formula when there are no random effects). Under the same assumption, another derivation as a limiting form of a parametrized maximum likelihood estimate was given by Laird [31].

When applied to the generalized linear stochastic model (1) in the augmented formulation discussed above, the REML objective function (6) takes the computationally most useful form given by Graser et al. [14].

The following proposition contains formulas for computing derivatives of the REML function. We write

$$\dot{\square} = \partial_\mu \square = \frac{\partial \square}{\partial \omega_\mu}$$

for the derivative with respect to a parameter ω_μ occurring in the covariance matrix.

Proposition [22, 32, 42, 50]. Let

$$\mathbf{P} = \mathbf{M} - \mathbf{M}\mathbf{A}\mathbf{B}^{-1}\mathbf{A}^T\mathbf{M} \tag{7}$$

where $\mathbf{A}$ and $\mathbf{B}$ are as previously defined and

$$\mathbf{M} = \mathbf{C}^{-1}$$

Then

$$\partial_\mu f = \operatorname{tr}\mathbf{P}_\varepsilon(\partial_\mu \mathbf{C}) \tag{8}$$

$$\partial_\mu \partial_\nu f = \mathbf{s}_\mu^T \mathbf{P} s_\nu + \delta_{\mu\nu} \tag{9}$$

where

$$\mathbf{P}_\varepsilon = \mathbf{P} - \mathbf{P}\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^T\mathbf{P} = \mathbf{P} - \mathbf{P}\mathbf{b}\mathbf{b}^T\mathbf{P} = \mathbf{P} - \mathbf{M}\mathbf{r}\mathbf{r}^T\mathbf{M} \tag{10}$$

$$s_\mu = (\partial_\mu \mathbf{C})\mathbf{M}\mathbf{r} \tag{11}$$

$$\delta_{\mu,\nu} = \operatorname{tr} \mathbf{P}_\varepsilon(\delta_\mu\delta_\nu\mathbf{C} - (\partial_\mu\mathbf{C})\mathbf{P}(\partial_\nu\mathbf{C})) \tag{12}$$

(Note that, since $\mathbf{A}$ is nonsquare, the matrix $\mathbf{P}$ is generally nonzero although it always satisfies $\mathbf{PA} = 0$.)

3. FULL AND INCOMPLETE ELEMENT FORMULATION

For the practical modeling of linear stochastic systems, it is useful to split model (3) into blocks of uncorrelated model equations which we call 'element equations'. The element equations usually fall into several types, distinguished by their covariance matrices. The model equation for an element ν of type γ has the form

$$\mathbf{A}_\nu\mathbf{x} = \mathbf{b}_\nu + \boldsymbol{\varepsilon}_\nu, \quad \operatorname{cov}(\boldsymbol{\varepsilon}_\nu) = \mathbf{C}_\gamma \tag{13}$$

Here $\mathbf{A}_\nu$ is the coefficient matrix of the block of equations for element number ν. Generally, $\mathbf{A}_\nu$ is very sparse with few rows and many columns, most of them zero, since only a small subset of the variables occurs explicitly in the νth element.

Each model equation has only one noise term. Correlated noise must be put into one element. All elements of the same type are assumed to have statistically independent noise vectors, realizations of (not necessarily Gaussian) distributions with zero mean and the same covariance matrix. (In our implementation, there are no constraints on the parametrization of the $\mathbf{C}_\gamma$, but it is not difficult to modify the formulas to handle more restricted cases.) Thus the various elements are assigned to the types according to the covariance matrices of their noise vectors.

3.1. *Example animal breeding applications*

In covariance component estimation problems from animal breeding, the vector $\mathbf{x}$ splits into small vectors $\boldsymbol{\beta}_k$ of (in our present implementation constant) size n_{trait} called 'effects'. The right-hand side $\mathbf{b}$ contains measured data vectors y_ν and zeros. Each index ν corresponds to some animal. The various types of elements are as follows.

Measurement elements: the measurement vectors $\mathbf{y}_\nu \in \mathbb{R}^{n_{\text{trait}}}$ are explained in terms of a linear combination of effects $\boldsymbol{\beta}_i \in \mathbb{R}^{n_{\text{trait}}}$,

$$\sum_{l=1}^{n_{\text{eff}}} \boldsymbol{\mu}_{\nu l}\boldsymbol{\beta}_{i_{\nu l}} = \mathbf{y}_\nu + \boldsymbol{\varepsilon}_\nu, \quad \operatorname{cov}(\boldsymbol{\varepsilon}_\nu) = \mathbf{C}_1$$

Here the $i_{\nu l}$ form an $n_{\text{rec}} \times n_{\text{eff}}$ index matrix, the $\boldsymbol{\mu}_{\nu l}$ form an $n_{\text{rec}} \times n_{\text{eff}}$ coefficient matrix, and the data records $\mathbf{y}_\nu^T$ are the rows of an $n_{\text{rec}} \times n_{\text{trait}}$ measurement matrix. In the current implementation, corresponding rows of the coefficient matrix and the

measurement matrix are concatenated so that a single matrix containing the floating point numbers results. If the set of traits splits into groups that are measured on different sets of animals, the measurement elements split accordingly into several types.

Pedigree elements: for some animals, identified by the index T of their additive genetic effect $\boldsymbol{\beta}_T$, we may know the parents, with corresponding indices V (father) and M (mother). Their genetic dependence is modeled by an equation

$$\frac{1}{2}\boldsymbol{\beta}_{V(\nu)} + \frac{1}{2}\boldsymbol{\beta}_{M(\nu)} - \boldsymbol{\beta}_{T(\nu)} = 0 + \boldsymbol{\varepsilon}_\nu, \quad \mathrm{cov}(\boldsymbol{\varepsilon}_\nu) = \mathbf{C}_2$$

The indices are stored in pedigree records which contain a column of animal indices $T(\nu)$ and two further columns for their parents $(V(\nu), M(\nu))$.

Random effect elements: certain effects $\boldsymbol{\beta}_{R(\gamma)}$ $(\gamma = 3, 4, \ldots)$ are considered as random effects by including trivial model equations

$$\boldsymbol{\beta}_{R(\gamma)} = 0 + \boldsymbol{\varepsilon}_\gamma, \quad \mathrm{cov}(\boldsymbol{\varepsilon}_\gamma) = \mathbf{C}_\gamma$$

As part of the model (13), these trivial elements automatically produce the traditional mixed model equations, as explained in section 2.

We now return to the general situation. For elements numbered by $\nu = 1, ..., N$, the full matrix formulation of the model (13) is the model (3) with

$$\mathbf{A} = \begin{pmatrix} A_1 \\ \vdots \\ A_N \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} b_1 \\ \vdots \\ b_N \end{pmatrix}, \quad \mathbf{C} = \begin{pmatrix} C_{\gamma(1)} & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & C_{\gamma(N)} \end{pmatrix}$$

where $\gamma(\nu)$ denotes the type of element ν.

A practical algorithm must be able to account for the situation that some components of $\mathbf{b}_\nu$ are missing. We allow for incomplete data vectors $\mathbf{b}$ by simply deleting from the full model the rows of $\mathbf{A}$ and $\mathbf{b}$ for which the data in $\mathbf{b}$ are missing. This is appropriate whenever the data are missing at random [43]; note that this assumption is also used in the missing data handling by the EM approach [6, 27].

Since dropping rows changes the affected element covariance matrices and their Cholesky factors in a nontrivial way, the derivation of the formulas for incomplete data must be performed carefully in order to obtain correct gradient information. We therefore formalize the incomplete element formulation by introducing projection matrices $\mathbf{P}_\nu$ coding for missing data pattern [31]. If we define $\mathbf{P}_\nu$ as the $(0,1)$ matrix with exactly one 1 per row (one row for each component present in $\mathbf{b}_\nu$), at most one 1 per column (one column for each component of $\mathbf{b}_\nu$), then $\mathbf{P}_\nu\mathbf{A}_\nu$ is the matrix obtained from $\mathbf{A}_\nu$ by deleting the rows for which data are missing, and $\mathbf{P}_\nu\mathbf{b}_\nu$ is the vector obtained from $\mathbf{b}_\nu$ by deleting the rows for which data are missing. Multiplication by $\mathbf{P}_\nu^T$ on the right of a matrix removes the columns corresponding to missing components. Conversely, multiplication by $\mathbf{P}_\nu^T$ on the left or $\mathbf{P}$ on the right restores missing rows or columns, respectively, by filling them with zeros.

Using the appropriate projection operators, the model resulting from the full element formulation (13) in the case of some missing data has the incomplete

element equations

$$\mathbf{P}_\nu \mathbf{A}_\nu \mathbf{x} = \mathbf{P}_\nu \mathbf{b}_\nu + \varepsilon'_\nu, \quad \mathrm{cov}(\boldsymbol{\varepsilon}'_\nu) = \mathbf{C}'_\nu \tag{14}$$

where

$$\mathbf{C}'_\nu = \mathbf{P}_\nu \mathbf{C}_{\gamma(\nu)} \mathbf{P}_\nu^T \tag{15}$$

The incomplete element equations can be combined to full matrix form (3), with

$$\mathbf{A} = \begin{pmatrix} P_1 A_1 \\ \cdot \\ \cdot \\ \cdot \\ P_N A_N \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} P_1 b_1 \\ \cdot \\ \cdot \\ \cdot \\ P_N b_N \end{pmatrix}, \quad \mathbf{C} = \begin{pmatrix} \mathbf{C}'_1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & C'_N \end{pmatrix} \tag{16}$$

and the inverse covariance matrix takes the form

$$\mathbf{M} = \begin{pmatrix} M_1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & M_N \end{pmatrix} \tag{17}$$

where

$$\mathbf{M}_\nu = \mathbf{C}_\nu^{'-1}$$

Note that $\mathbf{C}'_\nu$, $\mathbf{M}_\nu$, and $\log\det \mathbf{C}'_\nu$ (a byproduct of the inversion via a Cholesky factorization, needed for the gradient calculation) depend only on type $\gamma(\nu)$ and missing data pattern $\mathbf{P}_\nu$, and can be computed in advance, before the calculation of the restricted loglikelihood begins.

4. THE REML FUNCTION AND ITS GRADIENT IN ELEMENT FORM

From the explicit representations (16) and (17), we obtain the following formulas for the coefficients of the normal equations

$$\mathbf{a} = \sum_\nu (\mathbf{P}_\nu \mathbf{A}_\nu)^T \mathbf{M}_\nu (\mathbf{P}_\nu \mathbf{b}_\nu)$$

$$\mathbf{B} = \sum_\nu (\mathbf{P}_\nu \mathbf{A}_\nu)^T \mathbf{M}_\nu (\mathbf{P}_\nu \mathbf{A}_\nu)$$

After assembling the contributions of all elements into these sums, the coefficient matrix is factored into a product of triangular matrices

$$\mathbf{B} = \mathbf{R}^T \mathbf{R}$$

using sparse matrix routines [8, 20]. Prior to the factorization, the matrix is reordered by the multiple minimum degree algorithm in order to reduce the amount of fill in. This ordering needs to be performed only once, before the first function

evaluation, together with a symbolic factorization to allocate storage. Without loss of generality, and for the sake of simplicity in the presentation, we may assume that the variables are already in the correct ordering; our programs perform this ordering automatically, using the multiple minimum degree ordering 'genmmd' as used in 'Sparsepak' [43].

Note that $\mathbf{R}$ is the transposed Cholesky factor of $\mathbf{B}$. (Alternatively, one can obtain $\mathbf{R}$ from a sparse $\mathbf{QR}$ factorization of $\mathbf{A}$, see e.g. Matstoms [33].)

To take care of dependent (or nearly dependent) linear equations in the model formulation, we replace in the factorization small pivots $\leqslant \varepsilon \mathbf{B}_{ii}$ by 1. (The choice $\varepsilon = (\text{macheps})^{2/3}$, where macheps is the machine accuracy, proved to be suitable. The exponent is less than 1 to allow for some accumulation of roundoff errors, but still guarantees 2/3 of the maximal accuracy.) To justify this replacement, note that in the case of consistent equations, an exact linear dependence results in a factorization step as in the following

$$\begin{pmatrix} \times & & & & \mathbf{R} & & \\ & \times & & & & & \\ & & \times & \times & \times & \times & \times \\ & & \times & 0 & 0 & 0 & 0 \\ \mathbf{R}^T & & \times & & & & \\ & & \times & & & & \\ & & \times & & & & \end{pmatrix} \begin{pmatrix} \mathbf{y} \\ \\ \times \\ 0 \\ \\ \\ \end{pmatrix}$$

In the presence of rounding errors (or in case of near dependence) we obtain entries of order $\varepsilon \mathbf{B}_{ii}$ in place of the diagonal zero. (This even holds when $\mathbf{B}_{ii}$ is small but nonzero, since the usual bounds on the rounding errors scale naturally when the matrix is scaled symmetrically, and we may choose the scaling such that nonzero diagonal entries receive the value one. Zero diagonal elements in a positive semidefinite matrix occur for zero rows only, and remain zero in the elimination process.) If we add $\mathbf{B}_{ii}$ to $\mathbf{R}_{ii}$ when $\mathbf{R}_{ii} < \varepsilon \mathbf{B}_{ii}$ and set $\mathbf{R}_{ii} = 1$ when $\mathbf{B}_{ii} = 0$, the near dependence is correctly resolved in the sense that the extreme sensitivity or arbitrariness in the solution is removed by forcing a small entry into the ith entry of the solution vector, thus avoiding the introduction of large components in null space directions. (It is useful to issue diagnostic warnings giving the indices of the column indices i where such near dependence occurred.)

The determinant

$$\log \det \mathbf{B} = \log \det \mathbf{R}^T \mathbf{R} = 2 \sum \log |\mathbf{R}_{ii}|$$

is available as a byproduct of the factorization. The above modifications to cope with near linear dependence are equivalent to adding prior information on the distribution of the parameters with those indices where pivots changed. Hence, provided that the set of indices where pivots are modified does not change with the iteration, they produce a correct behavior for the restricted loglikelihood. If this set of indices changes, the problem is ill-posed, and would have to be treated by regularization methods such as ridge regression, which is far too expensive for the large-scale problems for which our method is designed. In practice we have not

seen a failure of the algorithm because of the possible discontinuity in the objective function caused by our procedure for handling (near) dependence.

Once we have the factorization, we can solve the normal equations $\mathbf{R}^T\mathbf{R}\mathbf{x} = \mathbf{a}$ for the vector $\mathbf{x}$ cheaply by solving the two triangular systems

$$\mathbf{R}^T\mathbf{y} = \mathbf{a} \quad \text{and} \quad \mathbf{R}\mathbf{x} = \mathbf{y}$$

(In the case of an orthogonal factorization one has instead to solve $\mathbf{R}\mathbf{x} = \mathbf{y}$, where $\mathbf{y} = \mathbf{Q}^T\mathbf{b}$.)

From the best estimate $\hat{\mathbf{x}}$ for the vector $\mathbf{x}$, we may calculate the residual as

$$\mathbf{r} = \mathbf{A}\hat{\mathbf{x}} - \mathbf{b} = \begin{pmatrix} r_1 \\ \vdots \\ r_N \end{pmatrix}$$

with the element residuals

$$\mathbf{r}_\nu = \mathbf{P}_\nu\mathbf{A}_\nu\hat{\mathbf{x}} - \mathbf{P}_\nu\mathbf{b}_\nu$$

Then we obtain the objective function as

$$f = \log\det\mathbf{R}^T\mathbf{R} + \sum_\nu \left(\mathbf{r}_\nu^T\mathbf{M}_\nu\mathbf{r}_\nu + \log|\det\mathbf{C}'_\nu|\right)$$

Although the formula for the gradient involves the dense matrix $\mathbf{B}^{-1}$, the gradient calculation can be performed using only the components of $\mathbf{B}^{-1}$ within the sparsity pattern of $\mathbf{R}^T + \mathbf{R}$. This part of $\mathbf{B}^{-1}$ is called the 'sparse inverse' of $\mathbf{B}$ and can be computed cheaply; cf. Appendix 1. The use of the sparse inverse for the calculation of the gradient is discussed in Appendix 2.

The resulting algorithm for the calculation of a REML function value and its gradient is given in *table I*, in a form that makes good use of dense matrix algebra in the case of larger covariance matrix blocks $\mathbf{C}_\gamma$. The symbol $\oplus$ denotes adding a dense subvector (or submatrix) to the corresponding entries of a large vector (or matrix). In the calculation of the symmetric matrices $\mathbf{B}'$, $\mathbf{W}$, $\mathbf{M}'$ and $\mathbf{K}'$, it suffices to calculate the upper triangle.

Symbolic factorization and matrix reordering are not present in *table I* since these are performed only once before the first function evaluation. In large-scale applications, the bulk of the work is in the computation of the Cholesky factorization and the sparse inverse. Using the sparse inverse, the work for function and gradient calculation is about three times the work for function evaluation alone (where the sparse inverse is not needed). In particular, when the number p of estimated covariance components is large, the analytic gradient takes only a small fraction $2/p$ of the time needed for finite difference approximations.

Note also that for a combined function and gradient evaluation, only two sweeps through the data are needed, an important asset when the amount of data is so large that it cannot be held in main memory.

Table I. Calculation of REML function value and gradient.

% preprocess covariance information
do for all types γ
  initialize $\mathbf{L}_\gamma$;
  $\mathbf{C}_\gamma = \mathbf{L}_\gamma \mathbf{L}_\gamma^T$;
  do for all elements ν of type γ
    % (a loop over the distinct missing data patterns is
    % sufficient, using a suitable reference list)
    $\mathbf{C}'_\nu = \mathbf{P}_\nu \mathbf{C}_\gamma \mathbf{P}_\nu^T$;
    $\mathbf{M}_\nu = \mathbf{C}_\nu'^{-1}$;
    $\lambda_\nu = \log |\det \mathbf{C}'_\nu|$;
  end
end

% assemble normal equations
$\mathbf{a} = 0$; $\mathbf{B} = 0$;
do for all types γ
do for all elements ν of type γ
    gather $\mathbf{P}_\nu \mathbf{A}_\nu$, $\mathbf{P}_\nu \mathbf{b}_\nu$ as dense matrix/vector
    $\mathbf{N} = (\mathbf{P}_\nu \mathbf{A}_\nu)^T \mathbf{M}_\nu$;
    $\mathbf{a}' = \mathbf{N}(\mathbf{P}_\nu \mathbf{b}_\nu)$; $\mathbf{a} = \mathbf{a} \oplus \mathbf{a}'$;
    $\mathbf{B}' = \mathbf{N}(\mathbf{P}_\nu \mathbf{A}_\nu)$; $\mathbf{B} = \mathbf{B} \oplus \mathbf{B}'$;
  end
end

% sparse factorization and sparse inverse
factorize $\mathbf{B} = \mathbf{R}^T \mathbf{R}$;
$f = 2 \sum \log |\mathbf{R}_{ii}|$; % $= \log \det \mathbf{R}^T \mathbf{R}$;
solve $\mathbf{R}^T \mathbf{y} = \mathbf{a}$ and $\mathbf{R}\mathbf{x} = \mathbf{y}$;
compute the sparse part of $\overline{\mathbf{B}}$;
% (overwrite factorization by sparse inverse)

% accumulate function value and gradient
do for all types γ
  $\mathbf{K} = 0$;
  do for all elements ν of type γ
    gather $[\overline{\mathbf{B}}]_\nu$, $\mathbf{P}_\nu \mathbf{A}_\nu$, $\mathbf{P}_\nu \mathbf{b}_\nu$ as dense matrices/vector
    $\mathbf{r} = (\mathbf{P}_\nu \mathbf{A}_\nu)\mathbf{x} - \mathbf{P}_\nu \mathbf{b}_\nu$;
    $f = f + \mathbf{r}^T \mathbf{M}_\nu \mathbf{r} + \lambda_\nu$;
    $\mathbf{W} = \mathbf{r}\mathbf{r}^T + ((\mathbf{P}_\nu \mathbf{A}_\nu)[\overline{\mathbf{B}}]_\nu)(\mathbf{P}_\nu \mathbf{A}_\nu)^T$;
    $\mathbf{M}' = \mathbf{M}_\nu(\mathbf{I} - \mathbf{W}\mathbf{M}_\nu)$;
    $\mathbf{K} = \mathbf{K} + \mathbf{P}_\nu^T \mathbf{M}' \mathbf{P}_\nu$;
  end
  $\partial f / \partial (\mathbf{L}_\gamma)_{ik} = 2(\mathbf{K}\mathbf{L}_\gamma)_{ik}$;
end

5. ANIMAL BREEDING APPLICATIONS

In this section we give a small numerical example to demonstrate the setup of various matrices, and give less detailed results on two large problems. Many other animal breeding problems have been solved, with similar advantages for the new algorithm as in the examples given below [1–3, 19, 38, 49].

5.1. Small numerical example

Table II gives the data used for a numerical example. There are in all eight animals which are listed with their parent codes in the first block under 'pedigree'. The first five of them have measurements, i.e. dependent variables listed under 'dep var'. Each animal has two traits measured except for animal 2 for which the second measurement is missing. Structural information for independent variables is listed under 'indep var'. The first column in this block denotes a continuous independent variable, such as weight, for which a regression is to be fitted. The following columns are some fixed effect, such as sex, a random component, such as herd and the animal identification. Not all effects were fitted for both traits. In fact, weight was only fitted for the first trait as shown by the model matrix in *table III*.

Table II. Data.

Pedigree			Indep var				Dep var	
1	6	7	90.1	1	1	1	10.5	790
2	1	7	87.3	1	2	2	13.2	–
3	1	4	93.5	2	1	3	12.6	881
4	2	1	88.6	2	3	4	14.4	751
5	3	4	91.8	2	4	5	12.0	834
0	–	–						
7	8	–						
8	–	–						

Pedigree: parents of entry in column one; indep var: independent variable; dep var: dependent variable.

Table III. Model matrix.

Effect	Trait 1	Trait 2
Weight	1	0
Sex	1	1
Herd	1	1
Animal	1	1

The input data are translated into a series of matrices given in *table IV*. To improve numerical stability, dependent variables are scaled by their standard deviation and mean, while the continuous dependent variable is shifted by its mean only.

Table IV. Derived matrices.

Dep var*		mtype	Addresses				Coefficients			
−0.98	−1.24	1	0	2	6	14	−0.16	1.00	1.00	1.00
−0.66	–	4	0	2	8	16	−2.96	1.00	1.00	1.00
0.38	0.59	1	0	4	6	18	3.24	1.00	1.00	1.00
1.54	1.00	1	0	4	10	20	−1.66	1.00	1.00	1.00
−0.27	−0.35	1	0	4	12	22	1.54	1.00	1.00	1.00
		2	14	24	26		1.41	−0.71	−0.71	
		2	16	14	26		1.41	−0.71	−0.71	
		2	18	14	20		1.41	−0.71	−0.71	
		2	20	16	14		1.41	−0.71	−0.71	
		2	22	18	20		1.41	−0.71	−0.71	
		2	24	0	0		1.00	0	0	
		2	26	28	0		1.15	−0.58	0	
		2	28	0	0		1.00	0	0	
		3	6				1.00			
		3	8				1.00			
		3	10				1.00			
		3	12				1.00			

Dep var*: (dependent variable − mean)/(standard deviation).

Since there is only one random effect (apart from the correlated animal effect), the full element formulation [13] has three types of model equations, each with an independent covariance structure $\mathbf{C}_\gamma$.

Measurement elements (type $\gamma = 1$): the dependent variables give rise to type $\gamma = 1$ as listed in the second column in *table IV*. The second entry is special in that it denotes the residual covariance matrix for this record with a missing observation. To take care of this, a new mtype is created for each pattern of missing values (with mtype = type if no value is missing) [20]; i.e. the different values of mtype correspond to the different matrices $\mathbf{C}'_\nu$. However, it is still based on $\mathbf{C}_1$ as given in *table V* which lists all types in this example.

Pedigree elements (type $\gamma = 2$): the next nine rows in *table IV* are generated from the pedigree information. With both parents known, three entries are generated in both the address and coefficient matrices. With only one parent known, two addresses and coefficients are needed, while only one entry is required if no parent information is available. For all entries the type is $\gamma = 2$ with the covariance matrix $\mathbf{C}_2$.

Random effect elements (type $\gamma = 3$): the last four rows in *table IV* are the entries due to random effects which comprise three herd levels in this example. They have type $\gamma = 3$ with the covariance matrix $\mathbf{C}_3$.

All covariance matrices are 2×2, so that $p = 3+3+3 = 9$ dispersion parameters need to be estimated.

The addresses in the following columns in *table IV* are derived directly from the level codes in the data (*table II*) allocating one equation for each trait within each level pointing to the beginning of first trait in the respective effect level. For convenience of programming the actual address minus 1 is used. For linear

Table V. Types of covariance matrices.

mtype	Type γ	Missing value
1	1	–
2	2	–
3	3	–
4	1	2

Table VI. Gradients.

γ	Gradient		
1	0.4099	−1.5086	−2.2794
2	0.1629	−0.2669	−0.7693
3	1.1274	−0.3431	−2.2794

Table VII. Solutions for dispersion parameters.

γ	(co)Variance components		
	$\widehat{\sigma}^2_{11}$	$\widehat{\sigma}_{12}$	$\widehat{\sigma}^2_{22}$
1	0.750	48.610	3149.049
2	0.735	−37.009	1863.490
3	+0.000	+0.000	0.004

covariables only one equation is created, leading to the address of 0 for all five measurements.

The coefficients corresponding to the above addresses are stored in another matrix as given in *table IV*. The entries are 1 for class effects and continuous variables in the case of regression (shifted by the mean).

The address matrices and coefficient matrices in *table IV* form a sparse representation of the matrix **A** of equation (3) and can thus be used directly to set up the normal equations. Note that only one pass through the model equations is required to handle data, random effects and pedigree information. Also, we would like to point out that this algorithm does not require a separate treatment of the numerator relationship matrix. Indeed, the historic problem of obtaining its inverse is completely avoided with this approach.

As an example of how to set up the normal equations, we look at line 12 of *table IV* (because it does not generate as many entries as the first five lines). For the animal labelled T in *table IV*, the variables associated with the two traits have index $T + 1$ and $T + 2$. The contributions generated from line 12,

$$(2\ 26\ 28\ 0\ 1.15\ -0.58\ 0)$$

are given in *table VIII*.

Table VIII. $\mathbf{A}_\nu$ corresponding to line 12 of *table IV*.

	27	28	29	30
27	$1.15 \times 1.15 \times 3$	$1.15 \times 1.15 \times (-0.0009)$	$1.15 \times (-0.58) \times 3$	$1.15 \times (-0.58) \times (-0.0009)$
28	$1.15 \times 1.15 \times (-0.0009)$	$1.15 \times 1.15 \times 3$	$1.15 \times (-0.58) \times (-0.0009)$	$1.15 \times (-0.58) \times 3$
29	$1.15 \times (-0.58) \times 3$	$1.15 \times (-0.58) \times (-0.0009)$	$(-0.58) \times (-0.58) \times 3$	$(-0.58) \times (-0.58) \times (-0.0009)$
30	$1.15) \times (-0.58) \times (-0.0009)$	$1.15 \times (-0.58) \times 3$	$(-0.58) \times (-0.58) \times (-0.0009)$	$(-0.58) \times (-0.58) \times 3$

Starting values for all $\mathbf{C}_\nu$ for the scaled data were chosen as $\frac{1}{3}$ for all variances and 0.0001 for all covariances, amounting to a point in the middle of the parameter space. With $\mathbf{C}_\nu$ specified as above we have for its inverse

$$\mathbf{M}_\nu = \begin{pmatrix} 3. & -0.0009 \\ -0.0009 & 3. \end{pmatrix}$$

Optimization was performed with a BFGS algorithm as implemented by Gay [12]. For the first function evaluation we obtain a gradient given in *table VI* with a function value of 17.0053530. Convergence was reached after 51 iterations with solutions given in *table VII* at a loglikelihood of 15.47599750.

5.2. A large problem

A large problem from the area of pig breeding has been used to test an implementation of the above algorithm in the VCE package [17]. The data set comprised 26 756 measurement records with six traits. *Table IX* gives the number of levels for each effect leading to 233 796 normal equations. The columns headed by 'trait' represent the model matrix (cf. *table III*) mapping the effects on the traits. As can be seen, the statistical model is different for the various traits.

Table IX. Structure of big problem.

Effect	T	Number of equations	Trait 1	Trait 2	Trait 3	Trait 4	Trait 5	Trait 6
Effect 1	C	6	1	1	1	1	0	0
Effect 2	C	6	0	0	0	0	1	1
Effect 3	F	504	1	1	1	1	0	0
Effect 4	F	12	1	1	1	1	1	1
Effect 5	F	114	1	1	1	1	1	1
Effect 6	F	30	1	1	1	1	0	0
Effect 7	F	3 090	0	0	0	0	1	1
Effect 8	R	50 256	1	1	1	1	1	1
Effect 9	A	179 778	1	1	1	1	1	1
Total no. eqns		233 796						

T: kind of effect with; C: fixed continuous effect; F: fixed class effect; R: random effect with covariance matrix $\mathbf{C}_3$; A: random effect with pedigree with covariance matrix $\mathbf{C}_2$.

Because traits 1 through 4 and traits 5 and 6 are measured on different animals no residual covariances can be estimated, resulting in two types 1a and 1b, with 4×4 and 2×2 covariance matrices $\mathbf{C}_{1a}$ and $\mathbf{C}_{1b}$. Together with the 6×6 covariance matrices $\mathbf{C}_2$ and $\mathbf{C}_3$ for pedigree effect 9 and random effect 8, respectively, a total of 55 covariance components have to be estimated. The coefficient matrix of the normal equations resulted in 3 961 594 nonzero elements in the upper triangle, which lead to 5 993 686 entries in the Cholesky factor.

We compared the finite difference implementation of VCE [17] with an analytic gradient implementation based on the techniques of the present paper. An unconstrained minimization algorithm written by Schnabel et al. [44] that approximates the first derivatives by finite differences was used to estimate all 55 components simultaneously. The run performed 37 021 function evaluations at 111.6 s each on a Hewlett Packard 755 model amounting to a total CPU time of 47.8 days. To our knowledge, it was the first estimate of more than 50 covariance components simultaneously for such a large data set with a completely general model. Factorization was performed by a block sparse Cholesky algorithm due to Ng and Peyton [41].

Using analytic gradients, convergence was reached after 185 iterations taking 13 min each; the less efficient factorization from Misztal and Perez-Enciso [39] was used here because of the availability of their sparse inverse code. An even slightly better solution was reached and only 41 h of CPU time were used, amounting to a measured speed-up factor of nearly 28. However, this speed-up underestimates the superiority of analytical gradients because the factorization used in the Misztal and Perez-Enciso's code is less efficient than Ng and Peyton's block sparse Cholesky factorization used for approximating the gradients by finite differences. Therefore, the following comparison will be based on CPU time measurements made on Misztal and Perez-Enciso's factorization code.

For the above data set the CPU usage of the current implementation – which has not yet been tuned for speed (so the sparse inverse takes three to four times the time for the numerical factorization) – is given in *table X*. As can be seen from this table computing one approximated gradient by finite differencing takes around $202.6 * 55 = 11\ 143$ s, while one analytical gradient costs only around four times the set-up and solving of the normal equations, i.e. 812 s. Thus, the expected speed-up would be around 14. The 37 021 function evaluations required in the run with approximated gradients (which include some linear searches) would have taken 86.8 days with the Misztal and Perez-Enciso code. Thus, the resultant superiority of our new algorithm is nearly 51 for the model under consideration. This is much larger than the expected speed-up of 14 mainly because, with approximated gradients, 673 optimization steps were performed as compared to the 185 with analytical gradients.

Table X. CPU timings per task and iteration.

Task	CPU time (s)
Assemble normal equations	81.64
Numerical factorization	118.45
Solving	2.49
Sparse inverse	470.11
Assembling gradients	129.44

Such a high number of iterations with approximated gradients could be observed in many runs with higher numbers of dispersion variables and can be attributed to the reduced accuracy of the approximated gradients. In some extreme cases, the

optimization process even aborted when using approximated gradients, whereas analytical gradients yielded correct solutions.

5.3. Further evidence

Table XI presents data on a number of different runs that have been performed with our new algorithm. The statistical models used in the datasets vary substantially and cover a large range of problems in animal breeding. The new algorithm showed the same behaviour also on a plant breeding dataset (beans) which has a quite different structure as compared to the animal data sets.

Table XI. Some data on runs with analytical gradients.

Dataset	Unknowns		nze (in millions)		Number of iterations
	Linear	Cov. Components	Coeff. matrix	Factor	
$Groel_s$	2908	2	0.03	0.03	19
$Groe3_s$	8724	12	0.28	0.31	60
*Duck*1	24713	2	0.11	0.11	18
DanaP	18674	9	0.16	0.23	31
$Groe8_s$	23264	72	1.82	2.91	138
$Groel_a$	181635	2	0.82	2.01	23
$Groe3_a$	544905	12	6.97	18.7	90
*Hung*1	233796	55	3.96	5.99	185
*Hung*2	257190	55	4.38	6.41	132
$Diel_a$	4240	3	0.02	0.03	25
$Die2_a$	8480	9	0.09	0.12	59
$Die3_a$	12720	18	0.21	0.28	79
$Die4_a$	16960	30	0.37	0.51	119
$Die5_a$	21200	45	0.57	0.79	163
$Die6_a$	25440	63	0.82	1.15	126
$Die7_a$	29680	84	1.11	1.55	104
$Die8_a$	33920	108	1.45	2.03	115
$Die9_a$	38160	135	1.83	2.58	177
Beans	7599	12	0.08	0.08	31

nze: number of nonzero elements (in millions); coeff. matrix: half stored coefficient matrix; factor: half stored Cholesky factor of the coefficient matrix.

The datasets (details can be obtained from the second author) cover a whole range of problem sizes both in terms of linear and covariance components. Accordingly, the number of nonzero elements varies substantially from a few ten thousands up to many millions. Clearly, the number of iterations increases with the number of dispersion variables with a maximum well below 200. Some of the runs estimated covariance matrices with very high correlations well above 0.9. Although this is close to the border of the parameter space it did not seem to slow down convergence, a behaviour that contrasts markedly with that of EM algorithms.

For the above datasets the ratio of obtaining the gradient after and relative to the factorization was between 1.51 and 3.69 substantiating our initial claim that the analytical gradient can be obtained at a small multiple of the CPU time needed to calculate the function value alone. (For the large animal breeding problem described in *table X*, this ratio was 2.96.) So far, we have not experienced any ratios that were above the value of 4. From this we can conclude that with increasing numbers of dispersion variables our algorithm is inherently superior to approximated gradients by finite differences.

In conclusion, the new version of VCE not only computes analytical gradients much faster than the finite difference approximations (with the superiority increasing with the number of covariance components), but also reduces the number of iterations by a factor of around three, thereby expanding the scope of REML covariance component estimation in animal breeding models considerably. No previous code was able to solve problems of the size that can be handled with this implementation.

ACKNOWLEDGEMENTS

Support by the H. Wilhelm Schaumann Foundation is gratefully acknowledged.

REFERENCES

[1] Brade W., Groeneveld E., Bestimmung genetischer Populationsparameter für die Einsatzleistung von Milchkühen, Arch. Anim. Breeding 2 (1995) 149–154.

[2] Brade W., Groeneveld E., Einfluß des Produktionsniveaus auf genetische Populationsparameter der Milchleistung sowie auf Zuchtwertschätzergebnisse, Arch. Anim. Breeding 38 (1995) 289–298.

[3] Brade W., Groeneveld E., Bedeutung der speziellen Kombinationseignung in der Milchrinderzüchtung. Züchtungskunde 68 (1996) 12–19.

[4] Chu E., George A., Liu J., Ng E., SPARSEPAK: Waterloo sparse matrix package user's guide for SPARSEPAK-A, Technical Report CS-84-36, Department of Computer Science, University of Waterloo, Ontario, Canada, 1984.

[5] Corbeil R., Searle S., Restricted maximum likelihood (REML) estimation of variance components in the mixed model, Technometrics 18, (1976) 31–38.

[6] Dempster A., Laird N., Rubin D., Maximum likelihood from incomplete data via the EM algorithm, J. Roy. Statist. Soc. B 39 (1977) 1–38.

[7] Ducos A., Bidanel J., Ducrocq V., Boichard D., Groeneveld E., Multivariate restricted maximum likelihood estimation of genetic parameters for growth, carcass and meat quality traits in French Large White and French Landrace Pigs, Genet. Sel. Evol. 25 (1993) 475–493.

[8] Duff I., Erisman A., Reid J., Direct Methods for Sparse Matrices, Oxford Univ. Press, Oxford, 1986.

[9] Erisman A., Tinney W., On computing certain elements of the inverse of a sparse matrix, Comm. ACM 18 (1975) 177–179.

[10] Fellner W., Robust estimation of variance components, Technometrics 28 (1986) 51–60.

[11] Fraley C., Burns P., Large-scale estimation of variance and covariance components, SIAM J. Sci. Comput. 16 (1995) 192–209.

[12] Gay D., Algorithm 611 – subroutines for unconstrained minimization using a model/trust-region approach, ACM Trans. Math. Software 9 (1983) 503–524.

[13] Gill J., Biases in balanced experiments with uncontrolled random factors, J. Anim. Breed. Genet. (1991) 69–79.

[14] Graser H., Smith S., Tier B., A derivative-free approach for estimating variance components in animal models by restricted maximum likelihood, J. Anim. Sci. 64 (1987) 1362–1370.

[15] Groeneveld E., Simultaneous REML estimation of 60 coariance components in an animal model with missing values using a Downhill Simplex algorithm, in: 42nd Annual Meeting of the European Association for Animal Production, Berlin, 1991, vol. 1, pp. 108–109.

[16] Groeneveld E., Performance of direct sparse matrix solvers in derivative free REML covariance component estimation, J. Anim. Sci. 70 (1992) 145.

[17] Groeneveld E. REML VCE – a multivariate multimodel restricted maximum likelihood (co)variance component estimation package, in: Proceedings of an EC Symposium on Application of Mixed Linear Models in the Prediction of Genetic Merit in Pigs, Mariensee, 1994.

[18] Groeneveld E., A reparameterization to improve numerical optimization in multivariate REML (co)variance component estimation, Genet. Sel. Evol. 26 (1994) 537–545.

[19] Groeneveld E., Brade W., Rechentechnische Aspekte der multivariaten REML Kovarianzkomponentenschätzung, dargestellt an einem Anwendungsbeispiel aus der Rinderzüchtung, Arch. Anim. Breeding 39 (1996) 81–87.

[20] Groeneveld E., Kovac M., A generalized computing procedure for setting up and solving mixed linear models, J. Dairy Sci. 73 (1990) 513–531.

[21] Groeneveld E., Csato L., Farkas J., Radnoczi L., Joint genetic evaluation of field and station test in the Hungarian Large White and Landrace populations, Arch. Anim. Breeding 39 (1996) 513–531.

[22] Harville D., Maximum likelihood approaches to variance component estimation and to related problems, J. Am. Statist. Assoc. 72 (1977) 320–340.

[23] Hemmerle W., Hartley H., Computing maximum likelihood estimates for the mixed A.O.V. model using the W transformation, Technometrics 15 (1973) 819–831.

[24] Henderson C., Estimation of genetic parameters, Ann. Math. Stat. 21 (1950) 706.

[25] Henderson C., Applications of Linear Models in Animal Breeding, University of Guelph (1984).

[26] Henderson C., Estimation of variances and covariances under multiple trait models, J. Dairy Sci. 67 (1984) 1581–1589.

[27] Jennrich R., Schluchter M., Unbalanced repeated-measures models with structured covariance matrices, Biometry 42 (1986) 805–820.

[28] Jensen J., Madsen P., A User's Guide to DMU, National Institute of Animal Science, Research Center Foulum Box 39, 8830 Tjele, Denmark, 1993.

[29] Kovac M., Derivative free methods in covariance component estimation, Ph.D. thesis, University of Illinois, Urbana-Champaign.

[30] Laird N., Computing of variance components using the EM algorithm, J. Statist. Comput. Simul. 14 (1982) 295–303.

[31] Laird N., Lange N., Stram D., Measures: Application of the EM algorithm, J. Am. Statist. Assoc. 82 (1987) 97–105.

[32] Lindstrom M., Bates D., Newton-Raphson and EM algorithms for linear mixed-effects models for repeated-measures data, J. Am. Statist. Assoc. 83 (1988) 1014–1022.

[33] Matstoms P., Sparse QR factorization in MATLAB, ACM Trans. Math. Software 20 (1994) 136–159.

[34] Meyer K., DFREML – a set of programs to estimate variance components under an individual animal model, J. Dairy Sci. 71 (suppl. 2) (1988) 33–34.

[35] Meyer K., Restricted maximum likelihood to estimate variance components for animal models with several random effects using a derivative free algorithm, Genet. Sel. Evol. 21 (1989) 317–340.

[36] Meyer K., Estimating variances and covariances for multivariate animal models by restricted maximum likelihood, Genet. Sel. Evol. 23 (1991) 67–83.

[37] Meyer K., Derivative-intense restricted maximum likelihood estimation of covariance components for animal models, in: 5th World Congress on Genetics Applied to Livestock Production, University of Guelph, 7–12 August 1994, vol. 18, 1994, pp. 365–369.

[38] Mielenz N., Groeneveld E., Müller J., Spilke J., Simultaneous estimation of covariances with REML and Henderson 3 in a selected chicken population, Br. Poult. Sci 35 (1994).

[39] Misztal I., Perez-Enciso M., Sparse matrix inversion for restricted maximum likelihood estimation of variance components by expectation-maximization, J. Dairy Sci. (1993) 1479–1483.

[40] Nelder J., Mead R., A simplex method for function minimization, Comput. J. 7 (1965) 308–313.

[41] Ng E., Peyton B., Block sparse Cholesky algorithms on advanced uniprocessor computers, SIAM J. Sci. Comput. 14 (1993) 1034–1056.

[42] Patterson H., Thompson R., Recovery of inter-block information when block sizes are unequal, Biometrika 58 (1971) 545–554.

[43] Rubin D., Inference and missing data, Biometrika 63 (1976) 581–592.

[44] Schnabel R., Koontz J., Weiss B., A modular system of algorithms for unconstrained minimization, Technical Report CU-CS-240-82 Comp. Sci. Dept., University of Colorado, Boulder, 1982.

[45] Spilke J., Groeneveld E., Comparison of four multivariate REML (co)variance component estimation packages, in: 5th World Congress on Genetics Applied to Livestock Production, University of Guelph, 7–12 August 1994, vol. 22, 1994, pp. 11–14.

[46] Spilke J., Groeneveld E., Mielenz N., A Monte-Carlo study of (co)variance component estimation (REML) for traits with different design matrices, Arch. Anim. Breed. 39 (1996) 645–652.

[47] Tholen E., Untersuchungen von Ursachen und Auswirkungen heterogener Varianzen der Indexmerkmale in der Deutschen Schweineherdbuchzucht, Schriftenreihe Landbauforschung Völkenrode, Sonderheft 111 (1990).

[48] Thompson R., Wray N., Crump R., Calculation of prediction error variances using sparse matrix methods, J. Anim. Breed. Genet. 111 (1994) 102–109.

[49] Tixier-Boichard M., Boichard D., Groeneveld E., Bordas A., Restricted maximum likelihood estimates of genetic parameters of adult male and female Rhode Island Red Chickens divergently selected for residual feed consumption, Poultry Sci. 74 (1995) 1245–1252.

[50] Wolfinger R., Tobias R., Sall J., Computing Gaussian likelihood and their derivatives for general linear mixed models, SIAM J. Sci. Comput. 15 (1994) 1294–1310.

APPENDIX 1: Computing the sparse inverse

A cheap way to compute the sparse inverse is based on the relation

$$\mathbf{R}\overline{\mathbf{B}} = \mathbf{R}^{-T} \tag{A1}$$

for the inverse $\overline{\mathbf{B}} = \mathbf{B}^{-1}$. By comparing coefficients in the upper triangle of this equation, noting that $(\mathbf{R}^{-1})_{ii} = (\mathbf{R}_{ii})^{-1}$, we find that

$$\sum_{j \geqslant i} \mathbf{R}_{ij}\overline{\mathbf{B}}_{jk} = \mathbf{R}_{ii}^{-1}\delta_{ik} \quad \text{for} \quad i \leqslant k$$

where δ_{ik} denotes the Kronecker symbol; hence

$$\overline{\mathbf{B}}_{ki} = \overline{\mathbf{B}}_{ik} = \mathbf{R}_{ii}^{-1}(\mathbf{R}_{ii}^{-1}\delta_{ik} - \sum_{j>i} \mathbf{R}_{ij}\overline{\mathbf{B}}_{jk}) \quad \text{for} \quad i \leqslant k \tag{A2}$$

To compute $\overline{\mathbf{B}}_{ik}$ from this formula, we need to know the $\overline{\mathbf{B}}_{jk}$ for all $j > i$ with $\mathbf{R}_{ij} \neq 0$. Since the factorization process produces a sparsity structure with the property

$$\mathbf{R}_{ij} \neq 0, \quad \mathbf{R}_{ik} \neq 0, \quad i \leqslant j \leqslant k \Rightarrow \mathbf{R}_{jk} \neq 0$$

(ignoring accidental zeros from cancellation that are treated as explicit zeros), one can compute the components of the inverse $\overline{\mathbf{B}}$ within the sparsity pattern of $\mathbf{R}^T + \mathbf{R}$ by equation (A2) without calculating any of its entries outside this sparsity pattern.

If equation (A2) is used in the ordering $i = n, n-1, \ldots, 1$, the only additional space needed is that for a copy of the $\mathbf{R}_{ij} \neq 0$, $(j > i)$, which must be saved before we compute the $\overline{\mathbf{B}}_{ik} (\mathbf{R}_{ik} \neq 0, k \geqslant i)$ and overwrite them over $\mathbf{R}_{ik}$. (A similar analysis is performed for the Takahashi inverse by Erisman and Tinney [9], based on an $\mathbf{LDL}^T$ factorization.) Thus the number of additional storage locations needed is only the maximal numbers of nonzeros in a row of $\mathbf{R}$.

The cost is a small multiple of the cost for factoring $\mathbf{B}$, excluding the symbolic factorization; the proof of this by Misztal and Perez-Enciso [39] for the sparse inverse of an $\mathbf{LDL}^T$ factorization applies almost without change.

APPENDIX 2: Derivation of the algorithm in *table I*

For the derivative with respect to a variable that occurs in $\mathbf{C}_\gamma$ only, equation (15) implies that

$$\dot{\mathbf{C}}'_\nu = \begin{cases} P_\nu \dot{C}_\gamma P_\nu^T & \text{if } \nu \text{ is an element of type } \gamma, \\ 0 & \text{otherwise.} \end{cases}$$

(The computation of $\dot{\mathbf{C}}_\gamma$ is addressed below.) Using the notation $[\ldots]_\nu$ for the νth diagonal block of $[\ldots]$ and $\mathrm{tr}\mathbf{P}^T\mathbf{X} = \mathrm{tr}\mathbf{X}\mathbf{P}^T$, we find from

$$\dot{f} = \mathrm{tr}\left[\mathbf{M} - \mathbf{M}\mathbf{r}\mathbf{r}^T\mathbf{M} - \mathbf{M}\mathbf{A}\mathbf{B}^{-1}\mathbf{A}^T\mathbf{M}\right]\dot{\mathbf{C}}$$

(a consequence of the Proposition) the formula

$$\dot{f} = \sum_{\gamma(\nu)=\gamma} \mathrm{tr}\mathbf{P}_\nu^T \left[\mathbf{M} - \mathbf{M}\mathbf{r}\mathbf{r}^T\mathbf{M} - \mathbf{M}\mathbf{A}\mathbf{B}^{-1}\mathbf{A}^T\mathbf{M}\right]_\nu \mathbf{P}_\nu \dot{\mathbf{C}}_\gamma$$

hence

$$\dot{f} = \sum_{\gamma(\nu)=\gamma} \mathrm{tr}\, \mathbf{P}_\nu^T \mathbf{M}'_\nu \mathbf{P}_\nu \dot{\mathbf{C}}_\gamma$$

with the symmetric matrices

$$\mathbf{K}'_\nu = \mathbf{M}_\nu - \mathbf{M}_\nu(\mathbf{r}_\nu\mathbf{r}_\nu^T + \mathbf{A}_\nu\mathbf{B}^{-1}\mathbf{A}_\nu^T)\mathbf{M}_\nu \tag{A3}$$

Therefore,

$$\dot{f} = \mathrm{tr}K_\gamma \dot{\mathbf{C}}_\gamma, \quad \text{where} \quad K_\gamma = \sum_{\gamma(\nu)=\gamma} \mathbf{P}_\nu^T \mathbf{M}'_\nu \mathbf{P}_\nu \tag{A4}$$

Up to this point, the dependence of the covariance matrix $\mathbf{C}_\gamma$ on parameters was arbitrary. For an implementation, one needs to decide on the independent parameters in which to express the covariance matrices. We made the following choice in our implementation, assuming that there are no constraints on the parametrization of the $\mathbf{C}_\gamma$; other choices can be handled similarly, with a similar cost resulting for the gradient. Our parameters are, for each type γ, the nonzero entries of the Cholesky factor $\mathbf{L}_\gamma$ of $\mathbf{C}_\gamma$, defined by the equation

$$\mathbf{C}_\gamma = \mathbf{L}_\gamma \mathbf{L}_\gamma^T$$

together with the conditions

$$(\mathbf{L}_\gamma)_{ik} = 0 \quad \text{if} \quad i < k, \quad (\mathbf{L}_\gamma)_{ii} > 0$$

since this automatically guarantees positive definiteness. (In the limiting case, where a block of the true covariance matrix is semidefinite only, this will be revealed in the minimization procedure by converging to a singular $\mathbf{L}_\gamma$ while each computed $\mathbf{L}_\gamma$ is still nonsingular.)

We now consider derivatives

$$\dot{\square} = \partial\square/\partial(\mathbf{L}_\gamma)_{ik}$$

with respect to the parameter

$$\omega_\mu = (\mathbf{L}_\gamma)_{ik}$$

where γ is one of the types, and the indices i, k satisfy $i \geqslant k$.

Clearly, $\dot{\mathbf{L}}_\gamma$ is zero except for a 1 in position (i, k), and, using the notation e^i for the ith column of an identity matrix, we can express this as

$$\dot{\mathbf{L}}_\gamma = e^i(e^k)^T$$

Therefore,

$$\dot{\mathbf{C}}_\gamma = (\mathbf{L}_\gamma\mathbf{L}_\gamma^T)^\bullet = \dot{\mathbf{L}}_\gamma\mathbf{L}_\gamma^T + \mathbf{L}_\gamma\dot{\mathbf{L}}_\gamma^T = e^i(e^k)^T\mathbf{L}_\gamma^T + \mathbf{L}_\gamma e^k(e^i)^T \tag{A5}$$

If we insert this into equation (A4), we find

$$\begin{aligned}\dot{f} &= \mathrm{tr}\mathbf{K}_\gamma \dot{\mathbf{C}}_\gamma = \mathrm{tr}\mathbf{K}_\gamma e^i (e^k)^T \mathbf{L}_\gamma^T + \mathrm{tr}\mathbf{K}_\gamma \mathbf{L}_\gamma e^k (e^i)^T \\ &= (e^k)^T \mathbf{L}_\gamma^T \mathbf{K}_\gamma e^i + (e^i)^T \mathbf{K}_\gamma \mathbf{L}_\gamma e^k = (\mathbf{L}_\gamma^T \mathbf{K}_\gamma)_{ki} + (\mathbf{K}_\gamma \mathbf{L}_\gamma)_{ik}\end{aligned}$$

so that

$$\partial f / \partial (\mathbf{L}_\gamma)_{ik} = 2(\mathbf{K}_\gamma \mathbf{L}_\gamma)_{ik} \tag{A6}$$

In order to make good use of the sparsity structure of the problem, we have to look in more detail at the calculation of $\mathbf{M}'_\nu$. The first interior term in $\mathbf{M}'_\nu$ is easy since

$$(\mathbf{r}_\nu \mathbf{r}_\nu^T)_{ij} = (\mathbf{r}_\nu)_i (\mathbf{r}_\nu)_j$$

Correct treatment of the other interior term is crucial for good speed. Suppose the ith row of $\mathbf{A}_\nu$ has nonzeros in positions $k \in \mathbf{I}_{\nu,i}$ only. Then the term of $\mathbf{K}'_\nu$ involving the inverse $\overline{\mathbf{B}} = \mathbf{B}^{-1}$ can be reformulated as

$$\begin{aligned}(\mathbf{A}_\nu \mathbf{B}^{-1} \mathbf{A}_\nu^T)_{ij} &= \sum_{k,l} (\mathbf{A}_\nu)_{ik} (\bar{\mathbf{B}})_{kl} (\mathbf{A}_\nu^T)_{lj} \\ &= \sum_{k \in \mathbf{I}_{\nu,i},\ l \in \mathbf{I}_{\nu,j}} (\mathbf{A}_\nu)_{ik} (\overline{\mathbf{B}})_{kl} (\mathbf{A}_\nu^T)_{lj} \\ &= \sum_{k \in \mathbf{I}_{\nu,i}, l \in \mathbf{I}_{\nu,j}} (\mathbf{A}_\nu)_{ik} ([\overline{\mathbf{B}}]_\nu)_{kl} (\mathbf{A}_\nu^T)_{lj}\end{aligned}$$

Hence $\mathbf{A}_\nu \mathbf{B}^{-1} \mathbf{A}_\nu^T$ is a product of small submatrices. Under our assumption that all entries of $\mathbf{C}_\gamma$ are estimated, $\mathbf{C}'_\nu$ and hence $\mathbf{M}_\nu$ and $[\overline{\mathbf{B}}]_\nu$ are structurally full. Therefore, $[\mathbf{R} + \mathbf{R}^T]_\nu$ is full, too, and $[\overline{\mathbf{B}}]_\nu$ is part of the sparse inverse and hence cheaply available. Since the factorization is no longer needed at this stage, the sparse inverse can be stored in the space allocated to the factorization.