Restricted maximum likelihood estimation of covariances in sparse linear models

by

Arnold Neumaier
Institut für Mathematik, Universität Wien
Strudlhofgasse 4, A-1090 Wien, Austria
email: neum@cma.univie.ac.at

and

Eildert Groeneveld [1]
Institut für Tierzucht und Tierverhalten
Bundesforschungsanstalt für Landwirtschaft
D-31535 Neustadt, Germany
email: eg@tzv.fal.de

February 1995, revised March 1996

Abstract. This paper surveys the theoretical and computational development of the restricted maximum likelihood (REML) approach for the estimation of covariance matrices in linear stochastic models.
A new derivation of this approach is given, valid under very weak conditions on the noise.
Then the calculation of the gradient of restricted loglikelihood functions is discussed, with special emphasis on the case of large and sparse model equations with a large number of unknown covariance components and possibly incomplete data. It turns out that the gradient calculations require hardly any extra storage, and only a small multiple of the number of operations needed to calculate the function values alone.
The analytic gradient procedure was integrated into the VCE package for covariance component estimation in large animal breeding models. It resulted in dramatic improvements of performance over the previous implementation with finite difference gradients. An example with more than 250 000 normal equations and 55 covariance components took hours instead of days of CPU time, and this was not an untypical case.

Keywords: restricted maximum likelihood, variance component estimation, missing data, sparse inverse, automatic differentiation

1991 MSC Classification: primary 62J10; secondary 65F20, 65F50

[1]Support by the H. Wilhelm Schaumann Foundation is gratefully acknowledged

1 Introduction

In animal breeding, selection of parents of the next generation is based on their predicted additive genetic value. Best linear unbiased prediction of genetic merit [29] requires the covariance structure of the model elements involved. In practical situations, these are usually unknown and must be estimated. During the last years restricted maximum likelihood (REML) [49, 27] has emerged as the method of choice in animal breeding for variance component estimation [40, 41, 42, 20, 21, 22, 52].

Initially, the expectation maximization (EM) algorithm (Dempster [8] was used for the optimization of the REML objective function [30, 53]. Applications were written for special cases, and Fellner [13] made large-scale applications possible through sparse matrix techniques. However, the slow (linear) convergence renders it a procedure for problems with relatively few covariance components only. A recent accelerated version using Aitken extrapolation and fast gradients (similar to those proposed here) performed well in a comparison by Misztal [45], though the numerical results given there estimate only few covariance components.
In 1987 Graser et al. [18] introduced the derivative free optimization, which in the following years led to the development of rather general computing algorithms and packages [40, 20, 34, 32] that were mostly based on the simplex algorithm of Nelder & Mead [47]. Kovac [34] made modifications that turned it into a stable algorithm that no longer converge to noncritical points, but this did not improve its inherent inefficiency for increasing dimensions. Ducos et al. [9] 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.
Lindstrom & Bates [37] used analytic formulas for first and second derivatives (no sparsity considerations), and pointed out that optimization on the Cholesky factor of the covariance matrices, together with Goldstein-Armijo line searches overcomes the robustness problems within a Newton-Raphson algorithm. Groeneveld [23] showed the efficiency of this procedure and implemented it in a general purpose package. A comparison of different packages [52] confirmed the general observation of Gill et al. [17] that simplex based optimization algorithms suffer from lack of stability, sometimes converging to noncritical points while the quasi-Newton procedure with optimization on the Cholesky factor was stable and much faster than any of the other general purpose algorithms. While this led to a speed-up of between 2 and (for some examples) 200 as compared to the simplex procedure, approximating gradients on the basis of finite differences was still exceedingly costly for higher dimensional problems [22].

It is well-known that optimization algorithms generally perform better with analytic gradients if the latter are cheaper to compute than finite difference approximations. General results on automatic differentiation (see, e.g., GRIEWANK & CORLISS [19]) imply that for every computable function it is possible to get recursively analytical gradients at the cost of a few function evaluations, and hence much cheaper than with finite differences. This is achieved at the cost of storage space of the order of the number of operations; but at some penalty on the evaluation cost, storage space can be held at a reasonable size. A common way of doing this is by differentiating the whole program by a package like ADIFOR [1]. However, ADIFOR could not cope with our animal breeding code and produced no derivative code. Moreover, standard arguments predict larger overhead in either storage or runtime when differentiating a sparse matrix package, compared to the approach presented below.

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. Because of the inherent sparsity of the equations in many applications, our implementation makes use of sparse matrix techniques, in particular sparse inverse calculations from a sparse Cholesky factorization. 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. A similar fast gradient technique has been proposed by MISZTAL & PEREZ-ENCISO [46] (see also THOMPSON et al. [54] for an improvement in its space complexity), using an LDL^T factorization and the Takahashi inverse [11]; no results in a REML application were given. FELLNER [14] precedes this development but is less efficient since he computes the full inverse columnwise, saving storage but not time.
A recent paper by FRALEY & BURNS [15] provides a different approach to the computation of function values and analytic gradients based on sparse matrix indefinite factorization techniques; no test results are given. Their derivation is in terms of the W transformation introduced by HEMMERLE & HARTLEY [28] and first used in a REML context by CORBEIL & SEARLE [6]. Other recent papers by WOLFINGER et al. [55] (based again on the W transformation) and MEYER [43] (based on the simpler REML objective formulation of GRASER et al. [18]) also provide this information (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) 2–4 times the work for a function evaluation. (In automatic differentiation terms, this is the gain in complexity expected for switching from forward differentiation to backward differentiation.) MEYER [43] used her analytic second derivatives in a Newton-Raphson algorithm for optimization. Because the optimization was not restricted to positive definite covariance matrix approximations, 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 it into the VCE covariance component estimation package for animal (and plant) breeding models (Groeneveld [22]). Here the gradient routine is combined it 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 (Spilke & Groeneveld [52]). Typical problem sizes in animal breeding applications are of the order of 200 000 least squares variables and perhaps 60 nonlinear variables in the covariance matrices. (The variables occuring linearly constitute breeding values and estimates of fixed and random effects.) But in some problem areas, millions of linear unknowns may have to be estimated, while an upper limit for the nonlinear parameters could be set to around 400.
In the past, the largest animal breeding problem ever solved ([25], 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 100MHz 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 hours of CPU time, with a speed-up factor of nearly 28 with respect to the finite difference approach. In the mean time, the new VCE implementation is being used world wide and has been applied successfully to hundreds of animal breeding problems, with comparable performance advantages [2, 3, 4, 24, 44, 38].

In Section 2 we fix notation for linear stochastic models and mixed model equations. In Section 3 we define the REML objective function, and review closed formulas for its gradient and Hessian. We then derive a new theorem that shows why minimization of the REML function can be expected to give good estimates for the covariance matrix under very weak conditions on the noise. In Sections 4–8 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 applies the method to a simple demonstration case and several large breeding models.

2 Linear stochastic models

In the following, A^T denotes the transposed matrix of A. The expression $\langle x \rangle$ denotes the expectation of a random variable x (or random vector, etc.). The statement

$$\varepsilon = \text{noise}(C)$$

expresses that the noise vector ε is a realization of a random vector with *covariance matrix*

$$\langle \varepsilon \varepsilon^T \rangle = C.$$

(Strictly speaking, the name 'covariance matrix' is appropriate only when also $\langle \varepsilon \rangle = 0$, which is usually assumed as well, but not really needed.) Several such statements are taken to imply the assumption that the respective random vectors are uncorrelated. The covariance matrix C is assumed to be symmetric and positive definite; in particular, the inverse exists and is symmetric and definite, too.

We shall consider the *linear stochastic model* in the simple form

$$Ax = b + \varepsilon, \quad \varepsilon = \text{noise}(C), \tag{1}$$

where $A \in \mathbb{R}^{m \times n}$, $x \in \mathbb{R}^n$, $b, \varepsilon \in \mathbb{R}^m$ and $C \in \mathbb{R}^{m \times m}$. The *normal equations* for the model (1) have the form

$$Bx = a, \tag{2}$$

where

$$B = A^T C^{-1} A,$$
$$a = A^T C^{-1} b.$$

By solving the normal equations (2), we obtain the best linear unbiased estimate (BLUE)

$$\hat{x} = B^{-1} a = B^{-1} A^T C^{-1} b \tag{3}$$

for the *state vector* x, and the noise $\varepsilon = Ax - b$ is estimated by the *residual*

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

More generally, many applications (including those to animal breeding) are based on the *generalized linear stochastic model*

$$y = X\beta + Zu + \eta, \quad u = \text{noise}(D), \ \eta = \text{noise}(G), \tag{4}$$

with *fixed effects* β and *random effects* u. Usually, D and G are block diagonal, with many identical blocks.
By combining the two noise terms, the model is seen to be equivalent to the simple model $y = X\beta + \eta', \eta' = \text{noise}(V)$, with the mixed model covariance

matrix $V = ZDZ^T + G$. Usually, V is huge and no longer block diagonal, leading to hardly manageable normal equations involving the inverse of V. However, HENDERSON [7] showed that the normal equations are equivalent to the mixed model equations

$$\begin{pmatrix} X^TG^{-1}X & X^TG^{-1}Z \\ Z^TG^{-1}X & Z^TG^{-1}Z + D^{-1} \end{pmatrix} \begin{pmatrix} \beta \\ u \end{pmatrix} = \begin{pmatrix} X^TG^{-1}y \\ Z^TG^{-1}y \end{pmatrix}. \tag{5}$$

This formulation avoids the inverse of the mixed model covariance matrix V and is the basis of most modern methods for solving (4).

FELLNER [12] observed that Henderson's mixed model equations are the normal equations of an augmented model in the simple form (1). The augmented model uses the state vector

$$x = \begin{pmatrix} \beta \\ u \end{pmatrix}$$

and the noise vector

$$\varepsilon = \begin{pmatrix} -\eta \\ u \end{pmatrix},$$

with resulting coefficient matrix

$$A = \begin{pmatrix} X & Z \\ 0 & I \end{pmatrix},$$

data vector

$$b = \begin{pmatrix} y \\ 0 \end{pmatrix},$$

and covariance matrix

$$C = \begin{pmatrix} G & 0 \\ 0 & D \end{pmatrix}.$$

Thus, without loss in generality, we may base our analysis and our algorithms on the simple model (1), with a covariance matrix 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 [28, 6, 55, 15].

3 Restricted loglikelihood

If the covariance matrix $C = C(\omega)$ contains unknown parameters ω, these can be estimated by minimizing the *restricted loglikelihood*

$$f := r^TC^{-1}r + \log\det C + \log\det B, \tag{6}$$

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 (6) depend on C and hence on ω.)
More precisely, (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 [27]). Under the same assumption, another derivation as a limiting form of a parametrized maximum likelihood estimate was given by LAIRD [36]. However, as we shall show below, minimizing the restricted loglikelihood is also justified when the distribution of the noise is unknown.
When applied to the generalized linear stochastic model (4) in the augmented formulation discussed above, the REML objective function (6) takes the computationally most useful form given by GRASER et al. [18].

In order to be able to use fast numerical optimization techniques, we need to derive a formula for the computation of its gradient. The following proposition contains the relevant derivative information; equivalent formulas have been given earlier by [27, 37, 55]. We write

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

for the derivative with respect to a parameter ω_μ occuring in the covariance matrix.
In the following, the symbols *tr* and *det* for traces and determinants take precedence over $+$, $-$ and $=$, but not over products and parentheses.

Proposition. Let

$$P = M - MAB^{-1}A^T M, \tag{7}$$

where

$$M := C^{-1}.$$

Then

$$P_\varepsilon := P - P\varepsilon\varepsilon^T P = P - Pbb^T P = P - Mrr^T M \tag{8}$$

and

$$\partial_\mu f = \operatorname{tr} P_\varepsilon(\partial_\mu C), \tag{9}$$

$$\partial_\mu \partial_\nu f = s_\mu^T P s_\nu + \delta_{\mu\nu}, \tag{10}$$

where

$$s_\mu = (\partial_\mu C)Mr, \tag{11}$$

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

Proof. Since $CM = I$, we have $\dot{C}M + C\dot{M} = \dot{I} = 0$, hence

$$\dot{M} = -M\dot{C}M.$$

Since

$$\dot{r}^T Mr = (A\hat{x}^\bullet)^T M(A\hat{x} - b) = (\hat{x}^\bullet)^T (A^T MA\hat{x} - A^T Mb) = 0,$$

we conclude that

$$(r^T Mr)^\bullet = \dot{r}^T Mr + r^T \dot{M}r + r^T M\dot{r} = r^T \dot{M}r = -r^T M\dot{C}Mr = -\operatorname{tr}(Mrr^T M\dot{C}).$$

Now,

$$(\log\det C)^\bullet = \operatorname{tr} C^{-1}\dot{C} = \operatorname{tr} M\dot{C}.$$

Since $\dot{B} = A^T\dot{M}A = -A^T M\dot{C}MA$, we have

$$(\log\det B)^\bullet = \operatorname{tr} B^{-1}\dot{B} = -\operatorname{tr} B^{-1}A^T M\dot{C}MA = -\operatorname{tr} MAB^{-1}A^T M\dot{C}.$$

Since $\dot{f} = (r^T Mr)^\bullet + (\log\det C)^\bullet + (\log\det B)^\bullet$, we find

$$\dot{f} = \operatorname{tr}\left[M - Mrr^T M - MAB^{-1}A^T M\right]\dot{C} \tag{13}$$

as expression for the derivative. Now

$$PA = MA - MAB^{-1}(A^T MA) = MA - MA = 0,$$

hence

$$PCP = P - PAB^{-1}A^T M = P \tag{14}$$

and

$$P\varepsilon = P(Ax - b) = -Pb = -Mb + MAB^{-1}A^T Mb = -Mb + MA\hat{x} = Mr.$$

This implies (8), and (9) follows directly using (13).

In order to derive (10), let $Q = B^{-1}A^T M$. Then $CP + AQ = I$, and by differentiation,

$$\dot{C}P + C\dot{P} + A\dot{Q} = 0, \text{ so that } P(\dot{C}P + C\dot{P}) = 0.$$

Since $A^T\dot{P} = (A^T P)^\bullet = 0$ by symmetry of P, we conclude that

$$\dot{P} = (I - MAB^{-1}A^T)\dot{P} = PC\dot{P} = -P\dot{C}P.$$

Therefore,

$$\begin{aligned}\dot{P}_\varepsilon &= \dot{P} - Pbb^T\dot{P} - \dot{P}bb^T P = -P\dot{C}P + Pbb^T\dot{C}P + P\dot{C}Pbb^T P \\ &= -P_\varepsilon\dot{C}P + P\dot{C}Pbb^T P,\end{aligned}$$

and differentiation of (9) with ν in place of μ gives

$$\begin{aligned}\partial_\mu\partial_\nu f &= \operatorname{tr} P_\varepsilon(\partial_\mu\partial_\nu C) + \operatorname{tr}(\partial_\mu P_\varepsilon)(\partial_\nu C)\\ &= \operatorname{tr} P_\varepsilon(\partial_\mu\partial_\nu C) - \operatorname{tr} P_\varepsilon(\partial_\mu \dot{C})P(\partial_\nu C) + \operatorname{tr} P(\partial_\mu C)Pbb^TP(\partial_\nu C)\\ &= \delta_{\mu\nu} + \operatorname{tr} Ps_\mu s_\nu^T,\end{aligned}$$

hence (10). □

Remarks. (i) A little work can be saved in the computation of P_ε (and numerical symmetry ensured) by noting that in terms of a Cholesky factorization

$$B = R^TR$$

we can write P_ε as

$$P_\varepsilon = M - ss^T - NN^T$$

where

$$s = Mr$$

and N is a solution of the triangular linear system

$$NR = MA.$$

(ii) After having computed P_ε, we can find the derivative of f with respect to any parameter in the covariance matrix C by differentiating C with respect to this variable and taking the trace (9). When B is a dense matrix or the number n of variables x_j in the linear model (1) is not too large ($n < 100$, say) then this formula for the derivative can be implemented directly. In this case, also the formula for the second derivatives is practical.
(iii) For large-scale problems, the computation of P_ε is very expensive, and improved techniques, such as those given in later sections, are needed for efficient gradient calculation.
(iv) Since (12) is expensive to compute and its expectation vanishes at ω^* (see the next proof), it is appropriate to use in place of (10) the cheaper approximation

$$f''(\omega) \approx S^TPS = S^TMS - U^TU,$$

where S is defined by its columns (11) and $U = R^{-T}(A^TMS)$ is obtained by multiple back substitution. This is a good initial Hessian approximation for a quasi-Newton method, and it can be recomputed aposteriori for estimating confidence intervals. Johnson & Thompson [33] use this approximation in a modified Newton method.

The proposition allows us to give a new and elegant derivation of the REML approach.

Theorem. *Let x and b be random vectors with residuals $\varepsilon = Ax - b$. If*

$$\langle \varepsilon \varepsilon^T \rangle = C(\omega^*) \tag{15}$$

then ω^* is a stationary point of

$$\Phi(\omega) := \langle f(\hat{x}(\omega), \omega) \rangle,$$

where $\hat{x}(\omega)$ is the solution of the normal equations

$$A^T C(\omega)^{-1} A\hat{x} = A^T C(\omega)^{-1} b$$

and

$$f(x, \omega) = (Ax - b)^T C(\omega)^{-1}(Ax - b) + \log\det C(\omega) + \log\det A^T C(\omega)^{-1} A.$$

Moreover, ω^* satisfies the second order necessary conditions for minimizers of Φ.

Proof. For $\omega = \omega^*$, (14) implies

$$\langle P_\varepsilon \rangle = \langle P - P\varepsilon\varepsilon^T P \rangle = P - PCP = 0.$$

Since C is independent of ε we find $\partial_\mu \Phi(\omega^*) = \langle \partial_\mu f(\omega^*) \rangle = 0$, so that $\langle f \rangle$ is stationary at ω^*. Moreover, $\langle \delta_{\mu\nu} \rangle = 0$, hence

$$\partial_\mu \partial_\nu \Phi(\omega^*) = \langle \partial_\mu \partial_\nu f(\omega^*) \rangle = s_\mu^T P s_\nu.$$

Thus the Hessian is $\Phi''(\omega^*) = S^T P S$, where S has the s_ν as columns. But as a Schur complement of the positive semidefinite matrix

$$\begin{pmatrix} A^T M A & A^T M \\ MA & M \end{pmatrix} = (A\ I)^T M (A\ I),$$

P and hence $S^T P S$ is positive semidefinite. Thus the second order necessary conditions hold. □

By the theorem, the optimal covariance parameter vector ω^* is a stationary point (and very likely a minimizer) of $\langle f(\hat{x}(\omega), \omega) \rangle$. Since in practice, we only have one particular realization of the random vector b, it is natural to estimate ω^* from this realization as a stationary point of $\hat{\omega}$ of $f(\hat{x}(\omega), \omega)$. Now, as one easily sees, $\hat{\omega}$ is also a stationary point of $f(x, \omega)$ as a function of x and ω (the original formulation of REML). Indeed, variation with respect to x only gives precisely the normal equations (2), and substitution of the solution into $f(x, \omega)$ gives the previous objective function.
Since, for fixed ω, the stationary point is in fact the global minimizer, it is natural to drop the expectation and calculate the estimate $\hat{\omega}$ as the global minimizer of

f, too. This is precisely the REML procedure. Of course, in practice, one usually only calculates a local minimizer, but in practice, the minimizer often seems to be unique; at least this is suggested by results obtained with repeated minimization from multiple starting points.
We emphasize that this justification of the REML approach, unlike in the traditional derivation of the REML approach, nowhere needs assumptions such as that the noise should be Gaussian. This is due to the fact that our derivation is not based on maximum likelihood arguments. Thus the REML method has now a similarly broad justification for covariance component estimation in linear models as the least square method (and its variation for general covariance matrices) is justified as best linear unbiased estimator (BLUE) by the Gauss-Markov theorem.

4 Full and incomplete element formulation

For the practical modeling of linear stochastic systems, it is useful to split a model (1) 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

$$A_\nu x = b_\nu + \varepsilon_\nu, \quad \varepsilon_\nu = \text{noise}(C_\gamma). \tag{16}$$

Here A_ν is the coefficient matrix of the block of equations for element number ν. Generally, A_ν 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 C_γ, 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.
For elements numbered by $\nu = 1, ..., N$, the full matrix formulation of the model (16) is the model (1) with

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

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

A practical algorithm must be able to account for the situation that some components of b_ν are missing. We allow for incomplete data vectors b by simply deleting

from the full model the rows of A and b for which the data in b are missing. This is appropriate whenever the data are missing at random (RUBIN [50]); note that this assumption is also used in the missing data handling by the EM approach (DEMPSTER et al. [8], p.11; JENNRICH & SCHLUCHTER [31]). However, in view of our distribution-free justification of REML given above, it is likely that the REML estimates computed with this deletion technique are reliable also in situations where the data are missing in a systematic, but measurement independent way. (An example from animal breeding is a sex limited trait like milk yield that can be measured only on females, if it is included in an analysis together with other traits that are measured on both sexes.)

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 done carefully in order to obtain correct gradient information. We therefore formalize the incomplete element formulation by introducing projection matrices P_ν coding for *missing data pattern* (LAIRD et al. [35]). If we define P_ν as the $(0,1)$ matrix with exactly one 1 per row (one row for each component present in b_ν), at most one 1 per column (one column for each component of b_ν), then $P_\nu A_\nu$ is the matrix obtained from A_ν by deleting the rows for which data are missing, and $P_\nu b_\nu$ is the vector obtained from b_ν by deleting the rows for which data are missing. Multiplication by P_ν^T on the right of a matrix removes the columns corresponding to missing components. Conversely, multiplication by P_ν^T on the left or 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 (16) in case of some missing data has the *incomplete element equations*

$$P_\nu A_\nu x = P_\nu b_\nu + \varepsilon'_\nu, \quad \varepsilon'_\nu = \text{noise}(C'_\nu) \tag{17}$$

where

$$C'_\nu = P_\nu C_{\gamma(\nu)} P_\nu^T. \tag{18}$$

The incomplete element equations can be combined to full matrix form (1), with

$$A = \begin{pmatrix} P_1 A_1 \\ \cdot \\ \cdot \\ \cdot \\ P_N A_N \end{pmatrix}, \quad b = \begin{pmatrix} P_1 b_1 \\ \cdot \\ \cdot \\ \cdot \\ P_N b_N \end{pmatrix}, \quad C = \begin{pmatrix} C'_1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & C'_N \end{pmatrix}. \tag{19}$$

and the inverse covariance matrix takes the form

$$M = \begin{pmatrix} M_1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & M_N \end{pmatrix}. \tag{20}$$

where

$$M_\nu = {C'_\nu}^{-1}.$$

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

5 The normal equations in element form

From the explicit representation (19), (20), we get the following formulas for the coefficients of the normal equations.

$$a = \sum_\nu (P_\nu A_\nu)^T M_\nu (P_\nu b_\nu),$$

$$B = \sum_\nu (P_\nu A_\nu)^T M_\nu (P_\nu A_\nu).$$

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

$$B = R^T R,$$

using sparse matrix routines [26, 10]. 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 need to be done only once, before the first function evaluation, together with doing 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 of course perform this ordering automatically, using the multiple minimum degree ordering *genmmd* as used in SPARSEPAK [5].
Note that R is the transposed Cholesky factor of B. (Alternatively, one can obtain R from a sparse QR factorization of A, see, e.g., MATSTOMS [39].)

To take care of dependent (or nearly dependent) linear equations in the model formulation, we replace in the factorization small pivots $\leq \varepsilon B_{ii}$ 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 case of consistent equations, an exact

linear dependence results in a factorization step looking like

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

In the presence of rounding errors (or in case of near dependence) we get entries of order εB_{ii} in place of the diagonal zero. (This even holds when 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 get 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 B_{ii} to R_{ii} when $R_{ii} < \varepsilon B_{ii}$ and set $R_{ii} = 1$ when $B_{ii} = 0$, 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

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

is available as a byproduct of the factorization. The above modifications to cope with near linear dependence amount 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 haven't 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 $R^T Rx = a$ for the state vector cheaply by solving the two triangular systems

$$R^T y = a \quad \text{and} \quad Rx = y.$$

(In case of an orthogonal factorization one has instead to solve $Rx = y$, where $y = Q^T b$.)

It turns out that, although the formula for the gradient involves B^{-1}, the gradient calculation can be done using the *sparse inverse* of $B = R^T R$ only, i.e., the components of B^{-1} within the sparsity pattern of $R^T + R$. This part of B^{-1} is called the *sparse inverse* of B. A cheap way to compute the sparse inverse is based on the relation

$$R\bar{B} = R^{-T} \tag{21}$$

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

$$\sum_{j \geq i} R_{ij} \bar{B}_{jk} = R_{ii}^{-1} \delta_{ik} \quad \text{for } i \leq k,$$

where δ_{ik} denotes the Kronecker symbol; hence

$$\bar{B}_{ki} = \bar{B}_{ik} = R_{ii}^{-1}(R_{ii}^{-1}\delta_{ik} - \sum_{j>i} R_{ij}\bar{B}_{jk}) \quad \text{for } i \leq k. \tag{22}$$

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

$$R_{ij} \neq 0,\ R_{ik} \neq 0,\ i \leq j \leq k \ \Rightarrow\ R_{jk} \neq 0$$

(ignoring accidental zeros from cancellation that are treated as explicit zeros), one can compute the components of the inverse $\bar{B}$ within the sparsity pattern of $R^T + R$ by (22) without calculating any of its entries outside this sparsity pattern. If (22) is used in the ordering $i = n, n-1, ..., 1$, the only additional space needed is that for a copy of the $R_{ij} \neq 0$, $(j > i)$, which must be saved before we compute the $\bar{B}_{ik}(R_{ik} \neq 0, k \geq i)$ and overwrite them over R_{ik}. (A similar analysis is done for the Takahashi inverse by Erisman & Tinney [11], based on an LDL^T factorization.) Thus the number of additional storage locations needed is only the maximal numbers of nonzeros in a row of R.
The cost is a small multiple of the cost for factoring B, excluding the symbolic factorization; the proof of this by Misztal & Perez-Enciso [46] for the sparse inverse of an LDL^T factorization applies almost without change. As described in the final part of the paper, measurements with our implementation confirm the low cost for a variety of animal breeding problems.

6 Function and gradient accumulation

Once we have the best estimate $\hat{x}$ for the state vector, we may calculate the residual as

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

with the element residuals

$$r_\nu = P_\nu A_\nu \hat{x} - P_\nu b_\nu.$$

Then we obtain the objective function as

$$f = \log\det R^T R + \sum_\nu \left(r_\nu^T M_\nu r_\nu + \log|\det C'_\nu| \right).$$

The calculation of the gradient is more involved. For the derivative with respect to a variable that occurs in C_γ only, (18) implies that

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

Using the notation $[...]_\nu$ for the νth diagonal block of $[...]$ and $\operatorname{tr} P^T X = \operatorname{tr} X P^T$, we find from (13) the formula

$$\dot{f} = \sum_{\gamma(\nu)=\gamma} \operatorname{tr} P_\nu^T \left[M - M r r^T M - M A B^{-1} A^T M \right]_\nu P_\nu \dot{C}_\gamma = \sum_{\gamma(\nu)=\gamma} \operatorname{tr} P_\nu^T M'_\nu P_\nu \dot{C}_\gamma$$

with the symmetric matrices

$$K'_\nu := M_\nu - M_\nu (r_\nu r_\nu^T + A_\nu B^{-1} A_\nu^T) M_\nu. \tag{23}$$

Therefore,

$$\dot{f} = \operatorname{tr} K_\gamma \dot{C}_\gamma, \quad \text{where } K_\gamma := \sum_{\gamma(\nu)=\gamma} P_\nu^T M'_\nu P_\nu. \tag{24}$$

Up to this point, the dependence of the covariance matrix C_γ 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 C_γ; 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 L_γ of C_γ, defined by the equation

$$C_\gamma = L_\gamma L_\gamma^T$$

together with the conditions

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

since this automatically guarantees positive definiteness.

We now consider derivatives

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

with respect to the parameter

$$\omega_\mu = (L_\gamma)_{ik},$$

where γ is one of the types, and the indices i, k satisfy $i \geq k$.
Clearly, $\dot{L}_\gamma$ 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

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

Therefore,

$$\dot{C}_\gamma = (L_\gamma L_\gamma^T)^\bullet = \dot{L}_\gamma L_\gamma^T + L_\gamma \dot{L}_\gamma^T = e^i (e^k)^T L_\gamma^T + L_\gamma e^k (e^i)^T. \tag{25}$$

If we insert this into (24), we find

$$\begin{aligned} \dot{f} &= \operatorname{tr} K_\gamma \dot{C}_\gamma = \operatorname{tr} K_\gamma e^i (e^k)^T L_\gamma^T + \operatorname{tr} K_\gamma L_\gamma e^k (e^i)^T \\ &= (e^k)^T L_\gamma^T K_\gamma e^i + (e^i)^T K_\gamma L_\gamma e^k = (L_\gamma^T K_\gamma)_{ki} + (K_\gamma L_\gamma)_{ik}, \end{aligned}$$

so that

$$\partial f / \partial (L_\gamma)_{ik} = 2 (K_\gamma L_\gamma)_{ik}. \tag{26}$$

In order to make good use of the sparsity structure of the problem, we have to look in more detail at the calculation of M'_ν. The first interior term in M'_ν is easy since

$$(r_\nu r_\nu^T)_{ij} = (r_\nu)_i (r_\nu)_j.$$

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

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

Hence $A_\nu B^{-1} A_\nu^T$ is a product of small submatrices. Under our assumption that all entries of C_γ are estimated, C'_ν and hence M_ν and $[\bar{B}]_\nu$ are structurally full. Therefore, $[R + R^T]_\nu$ is full, too, and $[\bar{B}]_\nu$ 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.
The resulting algorithm for the calculation of a REML function value and its gradient is given in Table 1, in a form that makes good use of dense matrix

Table 1: Calculation of REML function value and gradient

% preprocess covariance information
do for all types γ
  initialize L_γ;
  $C_\gamma = L_\gamma L_\gamma^T$;
  do for all elements ν of type γ
    % (a loop over the distinct missing data patterns is
    % sufficient if a suitable reference list is prepared)
    $C'_\nu = P_\nu C_\gamma P_\nu^T$;
    $M_\nu = C'^{-1}_\nu$;
    $\lambda_\nu = \log|\det C'_\nu|$;
  end
end

% assemble normal equations
$a = 0; B = 0$;
do for all types γ
  do for all elements ν of type γ
    gather $P_\nu A_\nu, P_\nu b_\nu$ as dense matrix/vector
    $N = (P_\nu A_\nu)^T M_\nu$;
    $a' = N(P_\nu b_\nu); a = a \bigoplus a'$;
    $B' = N(P_\nu A_\nu); B = B \bigoplus B'$;
  end
end

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

% accumulate function value and gradient
do for all types γ
  $K = 0$;
  do for all elements ν of type γ
    gather $[\bar{B}]_\nu, P_\nu A_\nu, P_\nu b_\nu$ as dense matrices/vector
    $r = (P_\nu A_\nu)x - P_\nu b_\nu$;
    $f = f + r^T M_\nu r + \lambda_\nu$;
    $W = rr^T + ((P_\nu A_\nu)[\bar{B}]_\nu)(P_\nu A_\nu)^T$;
    $M' = M_\nu(I - WM_\nu)$;
    $K = K + P_\nu^T M' P_\nu$;
  end
  $\partial f/\partial(L_\gamma)_{ik} = 2(KL_\gamma)_{ik}$;
end

algebra in case of larger covariance matrix blocks C_γ. The symbol $\bigoplus$ denotes adding a dense subvector (or submatrix) to the corresponding entries of a large vector (or matrix). In the calculation of the symmetric matrices B', W, M' and K', it suffices to calculate the upper triangle.
Symbolic factorization and matrix reordering are not present in Table 1 since these are done 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. As mentioned above, this implies that 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.

7 Animal breeding applications

In covariance component estimation problems from animal breeding, the state vector x splits into small vectors β_k of (in our present implementation constant) size n_{trait} called *effects*. The right-hand side 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 $y_\nu \in \mathbb{R}^{n_{trait}}$ are explained in terms of a linear combination of effects $\beta_i \in \mathbb{R}^{n_{trait}}$,

$$\sum_{l=1}^{n_{eff}} \mu_{\nu l} \beta_{i_{\nu l}} = y_\nu + \varepsilon_\nu, \quad \varepsilon_\nu = \text{noise}(C_1).$$

Here the $i_{\nu l}$ form an $n_{rec} \times n_{eff}$ index matrix, the $\mu_{\nu l}$ form an $n_{rec} \times n_{eff}$ coefficient matrix, and the *data records* y_ν^T are the rows of an $n_{rec} \times n_{trait}$ 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 β_T, we may know the parents, with corresponding indices V (father) and M (mother). Their genetic dependence is modeled by an equation

$$\frac{1}{2}\beta_{V(\nu)} + \frac{1}{2}\beta_{M(\nu)} - \beta_{T(\nu)} = 0 + \varepsilon_\nu, \quad \varepsilon_\nu = \text{noise}(C_2).$$

Table 2: 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
6	-	-						
7	8	-						
8	-	-						

pedigree: parents of entry in column one
indep var: independent variable
dep var: dependent variable

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 $\beta_{R(\gamma)}$ $(\gamma = 3, 4, ...)$ are considered as random effects by including trivial model equations

$$\beta_{R(\gamma)} = 0 + \varepsilon_\gamma, \quad \varepsilon_\gamma = \text{noise}(C_\gamma).$$

As part of the model (16), these trivial elements automatically produce the traditional mixed model equations, as explained in Section 2.

In the following we shall 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 [2, 3, 4, 24, 44, 38].

Small numerical example. Table 2 gives the data used for a numerical example. There are in all 8 animals which are listed with their parent codes in the first block under 'pedigree'. The first 5 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, like weight, for which a regression is to be fitted. The following columns are some fixed effect, like sex, a random component, like 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 3.

Table 3: 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 4. 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.

Since there is only one random effect, the full element formulation (16) has three types of model equations, each with an independent covariance structure C_γ.

Measurement elements (type $\gamma = 1$): the dependent variables give rise to type $\gamma = 1$ as listed in the second column in Table 4. 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) [26]; i.e., the different values of mtype correspond to the different matrices C'_ν. However, it is still based on C_1 as given in Table 5 which lists all types in this example.
Pedigree elements (type $\gamma = 2$): The next 9 rows in Table 4 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 C_2.
Random effect elements (type $\gamma = 3$): The last 4 rows in Table 4 are the entries due to random effects which comprise three herd levels in this example. They have type $\gamma = 3$ with the covariance matrix C_3.
All covariance matrices are 2×2, so that $p = 3 + 3 + 3 = 9$ nonlinear parameters need to be estimated.

The addresses in the following columns in Table 4 are derived directly from the level codes in the data (Table 2) allocating one equation for each trait within each level pointing to the beginning of first trait in the respective effect level. For linear covariables only one equation is created, leading to the address of 0 for all 5 measurements.

Table 4: 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)

Table 5: Types of covariance matrices

mtype	type γ	missing value
1	1	-
2	2	-
3	3	-
4	1	2

Table 6: 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 7: Solutions for nonlinear parameters

γ	(co)variance components		
	$\hat{\sigma_{11}}^2$	$\hat{\sigma_{12}}$	$\hat{\sigma_{22}}^2$
1	0.750	48.610	3149.049
2	0.735	-37.009	1863.490
3	+0.000	+0.000	0.004

The coefficients corresponding to the above addresses are stored in another matrix as given in Table 4. 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 4 form a sparse representation of the matrix A of (1) 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. As an example for how to set up the normal equations, we look at line 12 of Table 4 (because it does not generate as many entries as the first five lines, say). For the animal labelled T in Table 4, 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 8.

Table 8: A_ν corresponding to line 12 of Table 4

	27	28	29	30
27	$1.15 \times 1.15 \times 3$	$1.15 \times 1.15 \times (-.0009)$	$1.15 \times (-.58) \times 3$	$1.15 \times (-.58) \times (-.0009)$
28	$1.15 \times 1.15 \times (-.0009)$	$1.15 \times 1.15 \times 3$	$1.15 \times (-.58) \times (-.0009)$	$1.15 \times (-.58) \times 3$
29	$1.15 \times (-.58) \times 3$	$1.15 \times (-.58) \times (-.0009)$	$(-.58) \times (-.58) \times 3$	$(-.58) \times (-.58) \times (-.0009)$
30	$1.15) \times (-.58) \times (-.0009)$	$1.15 \times (-.58) \times 3$	$(-.58) \times (-.58) \times (-.0009)$	$(-.58) \times (-.58) \times 3$

Starting values for all C_ν for the scaled data were chosen $\frac{1}{3}$ for all variances and .0001 for all covariances, amounting to a point in the middle of the parameter space. With C_ν specified as above we have for its inverse

$$M_\nu = \begin{pmatrix} 3. & -.0009 \\ -.0009 & 3. \end{pmatrix}$$

Optimization was done with a BFGS algorithm as implemented by Gay [16]. For the first function evaluation we get a gradient given in Table 6 with a function value of 17.0053530. Convergence was reached after 51 iterations with solutions given in Table 7 at a loglikelihood of 15.47599750.

Table 9: Structure of big problem

Effect	T	no equ	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 eqn		233 796						

T: kind of effect with:
C: fixed continuous effect
F: fixed class effect
R: random effect with covariance matrix C_3
A: random effect with pedigree with covariance matrix C_2
no equ: number of equations

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 (GROENEVELD [22]). The data set comprised 26 756 measurement records with 6 traits. Table 9 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 3) mapping the effects on the traits. As can be seen, the statistical model is different for the various traits.
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 C_{1a} and C_{1b}. Together with the 6×6 covariance matrices C_2 and 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 [22] with an analytic gradient implementation based on the techniques of the present paper. An unconstrained minimization algorithm written by SCHNABEL ET AL. [51] 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 sec each on a Hewlett Packard 755 model amounting to a total CPU

Table 10: CPU timings per task and iteration

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

time of 47.8 days. To our knowledge, it was the first estimate of more than 50 covariance components simultaneously for such a big data set with a completely general model. Factorization was done by a block sparse Cholesky algorithm due to Ng & Peyton [48].
Using analytic gradients, convergence was reached after 185 iterations taking 13 minutes each; the less efficient factorization from Misztal & Perez-Enciso was used here because of the availability of their sparse inverse code [46]. An even slightly better solution was reached and only 41 hours 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 & Perez-Enciso's code is less efficient than Ng & 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 & 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 10. As can be seen from this table computing one approximated gradient by finite differencing takes around $202.6 * 55 = 11143$ seconds, while one analytical gradient costs only around four times the setup and solving of the normal equations, i.e., 812 seconds. Thus, the expected speedup would be around 14. The 37021 function evaluations required in the run with approximated gradients (which include some linear searches) would have taken 86.8 days with the Misztal & 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 speedup of 14 mainly because, with approximated gradients, 673 optimization steps were performed as compared to the 185 with analytical gradients.
Such a high number of iterations with approximated gradients could be observed in many runs with higher numbers of nonlinear unknowns 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

Table 11: Data on some runs with analytical gradients

	unknowns		nze (in millions)		number of
dataset	linear	non linear	coeff.matrix	factor	iterations
$Groe1_s$	2908	2	.03	.03	19
$Groe3_s$	8724	12	.28	.31	60
$Duck1$	24713	2	.11	.11	18
$DanaP$	18674	9	.16	.23	31
$Groe8_s$	23264	72	1.82	2.91	138
$Groe1_a$	181635	2	.82	2.01	23
$Groe3_a$	544905	12	6.97	18.7	90
$Hung1$	233796	55	3.96	5.99	185
$Hung2$	257190	55	4.38	6.41	132
$Die1_a$	4240	3	.02	.03	25
$Die2_a$	8480	9	.09	.12	59
$Die3_a$	12720	18	.21	.28	79
$Die4_a$	16960	30	.37	.51	119
$Die5_a$	21200	45	.57	.79	163
$Die6_a$	25440	63	.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	.08	.08	31

nze - number of non zero elements (in millions)
coeff.matrix - half stored coefficient matrix
factor - half stored Cholesky factor of the coefficient matrix

analytical gradients yielded correct solutions.

Another large problem. Another test was done on an even larger problem of a similar structure, again with 55 covariance components. For this problem, there were 257 190 normal equations and 4 380 522 and 6 405 934 nonzero elements in the coefficient matrix and factor, respectively. Convergence was reached here after 132 iterates. A direct comparison to the corresponding run based on finite differences is not possible, since (for reasons of time) we started the finite difference run not at the middle of the parameter space but closer to the (now known) solution. Nonetheless, the finite difference run needed 21 129 function evaluations in 384 optimization steps.

Further evidence. Table 11 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.
The data sets (details can be obtained from the second author) cover a whole range of problem sizes both in terms of linear and nonlinear unknowns. Accordingly, the number of nonzero elements vary substantially from a few ten thousends up to many millions. Clearly, the number of iterations increases with the number of nonlinear unknowns with a maximum well below 200. Some of the runs estimated covariance matrices with very high correlations well above .9. Although this is close to the boarder 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 10, 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 nonlinear unknowns 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.

References

[1] C. Bischof, A. Carle, P. Khamedi, A. Mauer, and P Hovland. ADIFOR 2.0 User's Guide, revision C. Technical Memorandum No. 192 CRPC-95516-S, Center for Research on Parallel Computation, Center for Research on Parallel Computation, 6100 S. Main Str. Rice University, Houston, TX 77005, USA (August 1995).

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

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

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

[5] E. Chu, A. George, J. Liu, and E. Ng. 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).

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

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

[8] A. P. N. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Stat. Soc. Ser. B* (1977), 1–38.

[9] A. Ducos, J. P. Bidanel, V. Ducrocq, D. Boichard, and E. Groeneveld. 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.

[10] I.S. Duff, A.M. Erisman, and J. K. Reid. Direct methods for sparse matrices. Oxford Univ. Press, Oxford (1986).

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

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

[13] W.H. Fellner. Sparse matrices and the analysis of variance by restricted maximum likelihood. In *Proc. Statistical Computing*, Alexandria, VA (1984). Amer. Stat. Assoc.

[14] W.H. Fellner. Sparse matrices and the analysis of variance components by likelihood methods. *Commun. Statist. Simul.* **16** (1987), 439–463.

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

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

[17] J. L. Gill. Biases in balanced experiments with uncontrolled random factors. *J. Animal Breeding Genetics* **108** (1991), 69–79.

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

[19] A. Griewank and G. F. Corliss. *Automatic Differentiation of Algorithms.* SIAM Publications, Philadelphia (1991).

[20] E. Groeneveld. Simultaneous REML estimation of 60 covariance components in an animal model with missing values using a Downhill Simplex algorithm. In *42nd Annual Meeting of the European Association for Animal Production*, Vol. 1, pp. 108–109, Berlin, Germany (1991).

[21] E. Groeneveld. Performance of direct sparse matrix solvers in derivative free REML covariance component estimation. *J. Animal Science* **70** (1992), 145.

[22] E. Groeneveld. 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* (E. Groeneveld, ed.) (1994).

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

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

[25] E. Groeneveld, L. Csato, J. Farkas, and L. Radnoczi. Multivariate Genetic Evaluation in the Hungarian Large White and Landrace Populations. (1995). submitted.

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

[27] D. A. Harville. Maximum likelihood approaches to variace component estimation and to related problems. *J. Amer. Statist. Assoc.* **72** (1977), 320–340.

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

[29] C.R. Henderon. *Applications of Linear Models in Animal Breeding*. University of Guelph (1984).

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

[31] R.I. Jennrich and M.D. Schluchter. Unbalanced repeated-measures models with structured covariance matrices. *Biometrics* **42** (1986), 805–820.

[32] J. Jensen and P. Madsen. *A user's guide to DMU*. National Institute of Animal Science, Research Center Foulum, Box 39, 8830 Tjele, Denmark (1993).

[33] D.L. Johnson and R. Thompson. Restricted maximum likelihood estimation of variance components for univariate animal models using sparse matrix techiques and average information. *J. Dairy Science* **78** (1995), 449–456.

[34] M. Kovac. *Derivative free methods in covariance component estimation*. PhD thesis, University of Illinois at Urbana-Champaign (1992).

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

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

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

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

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

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

[41] K. Meyer. 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.

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

[43] K. Meyer. Derivative-intense restricted maximum likelihood estimation of covariance components for Animal Models. In *5th World Congress on Genetics Applied to Livestock Production, Guelph*, Vol. 18, pp. 365 – 369 (1994).

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

[45] I. Misztal. Comparison of computing properties of derivative and derivative-free algorithms in variance-component estimation by REML. *J. Animal Breeding* **111** (1994), 346–355.

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

[47] J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal* **7** (1965), 308–313.

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

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

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

[51] R.B. Schnabel, J. E. Koontz, and B.E. Weiss. A modular system of algorithms for unconstrained minimization. Technical Report CU-CS-240-82, Comp. Sci. Dept., Univ. of Colorado at Boulder (1982).

[52] J. Spilke and E. Groeneveld. Comparison of four multivariate REML (co)variance component estimation packages. In *5th World Congress on Genetics Applied to Livestock Production, Guelph*, Vol. 22, pp. 11 – 14 (1994).

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

[54] R. Thompson, N.R. Wray, and R.E. Crump. Calculation of prediction error variances using sparse matrix methods. *J. Animal Breeding Genetics* **111** (1994), 102–109.

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