

Parallel Restricted Maximum Likelihood Estimation for Linear Models with a Dense Exogenous Matrix

J.M. Malard*

November 15, 2000

Abstract

Maximum likelihood estimates of covariance matrices for linear models occur in many statistical and stochastic applications such as estimating the genetic potential of cattle, financial time series analysis, the characterization of chemical mixtures and in general in the estimation of the parameters for stochastic differential equations. Restricted Maximum Likelihood (REML) is widely used in application areas where sampling bias is an important concern, but REML estimates are expensive to compute. Parallel implementations solely based on parallel dense matrix kernels need not scale well. This paper demonstrates that it is possible to compute estimates of covariance matrix for linear models based on restricted maximum likelihood (REML) efficiently on parallel computers. Two approaches to computing in parallel the gradient of the REML objective function are presented and compared. The covariance matrix is not assumed block diagonal. The implementations presented are based on PETSc and can run on any parallel computer supporting MPI.

1 Introduction

The solution of large linear systems $Ax = b$ is one of the main activity of the working numerical analyst. In this respect, structure is often very important. For instance, it is well known that the sparsity pattern of the matrix A , i.e. the structure of its non-zero elements, can play a crucial role in the efficient solution of large linear systems by either direct or iterative methods [12, 41]. The importance of the numerical structure of the coefficient matrix A has also attracted much attention: a well-known example being the multigrid preconditioners for heath flow equations. The numerical structure of the linear system can play yet another role in the context of over-determined systems of linear equations. Over-determined linear systems need not have a unique exact solution and are accurately written as in (1) where A has m rows and n columns with $m > n$ and where e is an unknown error vector. The choice of an optimal approximate solution x^* to (1) must be guided by some additional constraints arising from the application area. Such constraints are often expressed in terms of the error vector e or of the residual vector $r = -e$.

$$Ax = b + e \tag{1}$$

Quite commonly, the $m \times n$ *exogenous matrix* A and the vector x of *model parameters* may be assumed known exactly while the corresponding vectors e and b are tainted with uncertainty and may therefore be interpreted as realizations of some random vectors \mathbf{e} and \mathbf{b} . A major advantage of this approach is that knowledge about the structure of say \mathbf{e} can be incorporated into the solution of (1).

In this paper, the vector \mathbf{e} is assumed normally distributed with mean zero and with a positive definite $m \times m$ *variance-covariance matrix* C . Thus in the least squares approach to solving (1) the components of the covariance matrix C is assumed to be the identity matrix and one estimates the true vector x^* by

*Pacific Northwest National Laboratory, Battelle Boulevard, P.O. Box 999, Richland, WA 99352. The Pacific Northwest National Laboratory is operated by Battelle Memorial Institute for the U.S. Department of Energy under Contract DE-AC06-76RLO 1830.

the vector \hat{x} that minimizes the Euclidean norm of e , see for example [17]. In the General Gauss-Markov Linear Model, the matrix C is assumed known, symmetric and positive definite. Dependencies among the various elements of the random vector \mathbf{e} are reflected in both the sparsity and the numerical structure of the covariance matrix C . Thus in many applications, the covariance matrix C is not known exactly but its structure is known partially. For example, C may be known up to a scalar multiple, as in $C = \sigma^2\Omega$ where Ω is a known symmetric positive-definite matrix. In other cases, the non-zero entries of the Cholesky factor L of C may be parameterized by a number of *distribution parameters* $\Theta = (\theta_1, \dots, \theta_q)^T$. In all cases, values Θ^* of Θ and x^* of x are sought that optimizes a relevant objective function. Since the optimal covariance matrix $C^* = C(\Theta^*)$ need not have a closed form expression, an iterative process takes place where an initial guess $C^{[0]} = C(\Theta^{[0]})$ is refined until convergence is reached or some failure condition is met. The exact realization e of the random vector \mathbf{e} being unknown, its actual value is replaced by an appropriate estimate. It is customary to estimate the realization e of the error vector by $\tilde{e} = Ax - b$ where \tilde{x} is the Best Linear Unbiased Estimate (BLUE) for x , that is: \tilde{x} is the solution of the normal equations (2) below.

$$A^T C^{-1} Ax = A^T C^{-1} b \quad (2)$$

Some standard choices of objective function for computing optimal covariance matrices have appeared over the years. The principle of Maximum Likelihood is employed in many statistical and optimization problems, for example to compute semi-variograms in Geostatistics [48], to calibrate subsurface flow models and analyze the impact of uncertainty on the output of these models [7], to estimate the parameters of stochastic differential equations that model financial time series [18] and to compute confidence intervals for the parameters of Hidden Markov models [1] to name a few applications. In Maximum Likelihood estimation as the name implies one maximizes the likelihood function L of the error vector e as a function of the vector Θ . This function for the linear model (1) is shown in Equation (3), see for example [27].

$$L(\Theta) = (2\pi)^{-\frac{q}{2}} |C(\Theta)|^{-\frac{1}{2}} e^{-\frac{\tilde{e}^T C(\Theta)^{-1} \tilde{e}}{2}} \quad (3)$$

The likelihood is well defined when the matrix C is non-singular. Nonetheless, the determinant $|C|$ on the right-hand side of (3) can be arbitrarily large or small. It is therefore customary not to maximize the likelihood L but to minimize a linear function Φ of the logarithm of the likelihood as shown in Equation (4). The logarithmic terms on the right hand side of Equation (4) can be interpreted as a penalty term added to the residual sum of squares term $\tilde{e}^T C^{-1} \tilde{e}$.

$$\Phi(\Theta) = \log |C| + \tilde{e}^T C^{-1} \tilde{e} \quad (4)$$

Patterson and Thompson [39] proposed a general Restricted Maximum Likelihood (REML) procedure which is less sensitive to sampling bias than Maximum Likelihood and is consistent with ANOVA when the matrix C is diagonal. REML estimation differs from Maximum Likelihood by an additional logarithmic term in its objective function Ψ as shown in Equation (5) below. This objective function Ψ can be derived from the log-likelihood of the so-called error contrasts, see [19], [16] or [34] for details.

$$\Psi(\theta_1, \dots, \theta_q) = \log |C| + \log |A^T C^{-1} A| + r^T C^{-1} r \quad (5)$$

This additional term makes REML estimates of covariance matrices potentially much more expensive to compute than ML estimates. Nonetheless, REML estimation has become the method of choice in areas such as dairy production and heritability studies, see for example [14, 29]. REML estimation has also been used in the context of crop quality studies [43], pollution monitoring [4, 46], analysis of hydrological data [9]. Since the REML objective function Ψ differs from the ML objective function Φ by a single additional term, the algorithms presented in this paper can be easily tailored to compute ML estimates of variance and covariance components.

A spectrum of local optimization algorithms can be applied to minimize the REML objective function Ψ and many algorithms and packages are available for this purpose on sequential computers, see for example the excellent review in [34]. Several of these packages rely on a Newton-Raphson, a modified Newton or a Quasi-Newton approach. The Hessian matrix of the REML objective function Ψ is dense and its size is quadratic in the number of distribution parameters. This in itself limits the number of

distribution parameters $\theta_1 \dots \theta_q$ that can be estimated effectively using a Newton method. However, even small Hessian matrices and gradient vectors of Ψ can be expensive to compute exactly. Tradeoffs are typically made between computing inexpensive approximations of Hessian matrices and gradient vectors and minimizing the number of iterations required to reach an optimum value of Θ . Assumptions are often made about the structure of the covariance matrix in order to further speedup the computation of the gradient. Thus Neumaier and Groenvelde [34] rely on a block diagonal structure of the covariance matrix C to speed up the computation of either approximate or exact gradients. Mitztal [31] uses such analytical gradients to speedup a REML algorithm based on an accelerated Expectation Maximization approach.

Parallel optimization algorithms are an active area of research, see for example the reviews by Schnabel [42], Pflug and Świątanowski [40]. In particular, the solution of the Generalized Linear Model on parallel computers is an active field of research, as exemplified by the recent monograph by Kontoghiorghes on parallel Linear Model Estimation [23]. Matsuda et al. [26] and Ceron et al [8] investigate hill-climbing derivative-free algorithms for maximizing the likelihood of phylogenetic trees from DNA sequences. Jones et al. [21] and Möller [32] describe parallel Maximum Likelihood algorithms specialized for positron emission tomography. Of particular relevance to the present paper, Bull et al. [6, 5] report recent work on parallel Maximum Likelihood in the context of hierarchical models using parallel dense matrix algebra kernels on Symmetric Memory Multiprocessors (SMP). In the latter work, special care is taken to compute in parallel only those dense matrix operations that have enough arithmetic contents relative to the synchronization costs. Linear algebra kernels such as those provided via the Basic Linear Algebra Subroutines (BLAS) or LAPACK are essential tools for optimizing the performance of serial software, see for example [11]. Such interfaces typically provide the first line of attack when parallelizing existing software. However, in the case of REML estimation, the percentage of time spent within these linear algebraic interfaces can range from very large to very small depending on the parameters of the linear model of interest. The present paper demonstrates that REML estimation is a numerically intensive procedure that is best implemented as a mixture of dense and sparse linear algebra kernels at level of the covariance matrix C , i.e. Level 3 in [6]. With the notable exception of Mitztal [30], it appears that little attention has been given to general purpose REML estimation on supercomputers despite its large computational cost and practical interest.

A serious impediment to the development of REML software on supercomputers has been that efficient serial implementations of gradient computations in REML estimation typically rely on both the existence of a block diagonal structure of the covariance matrix and on complex data structures. In application areas such as livestock and crop management the covariance matrix is assumed block diagonal, reflecting independent trials. This assumption simplifies the computation of $\nabla \Psi$ because then C and C^{-1} share the same block diagonal structure. The size of diagonal blocks is typically small since for a dense diagonal block, the number of covariance components is quadratic in the number of random effects in the statistical model. One possibility for increasing the block sizes is to take advantage of the sparsity structure of each block, see for example [15]. On the other hand, assumptions made about the covariance structure of field data can result in under-estimates of the variability of the spatio-temporal data, see for example Meiring, Guttorp and Sampson [28]. Similarly, banded covariance matrices occur in financial applications. A general approach is thus needed to estimate efficiently sparse covariance matrices whose inverses are too large to fit in the local memory of a single process. The purpose of this paper is to show this is possible using publicly available parallel sparse linear algebra and optimization software.

The covariance matrix C is parameterized here by the non-zero entries of its lower-triangular Cholesky factor L , i.e. $C = L^T L$. The sparsity pattern of the matrix L is otherwise arbitrary and remains unchanged throughout the computation. The exogenous matrix A is assumed dense with full column rank and is independent of the vector Θ of distribution parameters. Dependant columns of A can be removed before the start of REML estimation. Many details of practical REML estimation are left aside in this paper such as the issue of missing data is not addressed although it is important from an application perspective. It should be noted in this respect that the algorithms presented here can easily be adapted for a matrix L whose sparsity pattern evolves during the computation. Another omission is the absorption of effects introduced by Thompson [44, 45] that can significantly reduce the size of inverted or factored matrices. The resulting “transformed” covariance matrix typically has many small diagonal blocks, which in turn reduces the effectiveness of parallelization strategies based on parallel dense matrix kernels. Constraints on the distribution and model parameters are not considered except

for the non-negative definiteness of C imposed by its Cholesky factorization. Actual positive definiteness could be enforced via some barrier function or by inequality constraints on the elements of the vector Θ as for example in [10].

The parallel implementations presented in this paper are made using the Portable, Extensible Toolkit for Scientific Computation (PETSc) [2]. PETSc was developed as an object oriented toolkit for solving partial differential equations in large-scale scientific applications. This package also supports unconstrained minimization and the solution of systems of non-linear equations, both using line search and trust region methods. PETSc is built on top of the Message Passing Interface (MPI); it is portable and has extensive debugging and benchmarking capabilities compared to other possible approaches. The default matrix distribution is by blocks of contiguous rows. The parallel linear system solvers in PETSc are Krylov methods; no parallel direct solver for distributed linear systems or parallel incomplete factorization preconditioner is included in the PETSc distribution. Despite these limitations, the recourse to a canned toolkit such as PETSc has considerably reduced the implementation time of the algorithms presented here. Performance measurements and verification runs were performed on a 512 processor IBM SP with Power II nodes at the Molecular Science Computing Facility (MSCF) in the William R. Wiley Environmental Molecular Sciences Laboratory at the Pacific Northwest National Laboratory, on the Cray T3E at the US National Energy Research Scientific Computing Center and on a 2 processor SGI Octane.

The paper is organized as follows. Restricted Maximum Likelihood is reviewed in Section 2 where the notation is set. A parallel algorithm for computing the REML objective function is presented in Section 3. Two algorithms for computing the gradient of the REML objective function are presented and compared in Section 4. One is based on mixture of structured adjoint and tangent computations and the other is based on Cramer’s Formula. Numerical experiments and scalability figures are presented in Section 5.

2 REML Covariance Matrix Estimation

2.1 Assumptions and Notation

Some objects such as the gradient of the optimization function are viewed both as vectors and as matrices. For any matrix $X \in \mathfrak{R}^{r \times s}$, $\text{vec}(X)$ denotes the vector of length rs obtained by stacking the columns of the matrix X . The transpose of the matrix X is denoted X^T . Array indices start at 1 and the Fortran 90 notation for array indices and sections is followed. If Q denotes a matrix, then the i^{th} column of Q is denoted either $Q(:, i)$ or Q_i . The element of Q in row i and column j is denoted either $Q(i, j)$ or q_{ij} .

Throughout the paper, the letter p denotes the number of active processes in the computation. This number is constant. The exogenous matrix $A \in \mathfrak{R}^{m \times n}$ is assumed dense, over determined and of full column rank, in particular $m > n$. It is assumed throughout the paper that the matrix $B = L^{-1}A$ is such that for a QR factorization $B = QR$ with Q orthogonal and $R \in \mathfrak{R}^{n \times n}$ upper triangular, the matrix R is small enough to fit in the local memory of each process. This assumption can be relaxed provided the latency of remote memory accesses is not too large. The covariance matrix C admits a parameterization of the form $C = LL^T$ where L is a lower triangular matrix whose entries depend linearly on the unknown vector $\Theta = (\theta_1, \dots, \theta_q)^T$ that parameterizes the normal distribution of the zero mean error vector e . Without loss of generality, the non-zero entries of the matrix L are components of the vector Θ . The vector Θ is assumed small enough to be replicated onto all p processes.

It is assumed here for simplicity that no component of Θ appears more than once in any row of L . In this way, the position of each of the parameters $\theta_1, \dots, \theta_q$ within the sparse matrix L can be recorded in a single $m \times q$ integer valued matrix \tilde{L} . The entries of the matrix \tilde{L} are defined by: $\tilde{L}_{i,k} = j$ when $L_{i,j} = \theta_k$ with all other entries of \tilde{L} being zero. The matrix \tilde{L} is used in two situations: to reset the matrix L when the vector Θ is updated and to compute the product of a vector by a partial derivative of either L or its transpose. The former operation has little effect on the overall performance of REML estimation. The latter operation is simply a vector scatter operation, i.e. a redistribution of parts of this vector. Multiple appearances of an element of Θ within the same row of L might also be recorded into a structural matrix \hat{L} that has the same sparsity structure as L but whose entries point to the appropriate element of Θ . The matrix L is distributed in the same manner as the matrix \tilde{L} onto the active processes. This arrangement ensures that since the vector Θ is known to all processes, each process can compute its

local portion of L without further need for communication. The computation of L^T on the other hand requires communication among processes.

2.2 The REML optimization problem

The linear system (1) can be rewritten as in (6) where $Cu = -r$. For a given matrix C , this last system has m equations and $m+n$ unknowns. It will, if consistent, be satisfied by infinitely many pairs of vectors x and u .

$$Ax + Cu = b \tag{6}$$

This indeterminacy is frequently resolved in statistics by minimizing the norm of $v = -L^{-1}r = L^T u$, see for example [17]. The system (6) can be combined with the normal equations (2) to yield the augmented linear system (7) below. C.C. Paige [37] has shown how this system can be solved accurately when the matrix C is only semi-definite using his generalized QR factorization. There are cases when for large values of m , the resulting $m \times m$ orthogonal factor of C may be dense. Here it is assumed that the matrix C is invertible and that the matrix A has full column rank which in turns implies that the augmented system (7) is non-singular.

$$\begin{pmatrix} C & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} u \\ x \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix} \tag{7}$$

One nice aspect of the augmented system (7) is that given an estimate for C^* the vector u can be estimated independently from the BLUE estimate \tilde{x} of x^* . The latter estimate can be computed from the optimizer C^* by solving the Normal Equations (2). The augmented system (7) also yields a simple expression for the partial derivatives of u with respect to the model parameters Θ . These partial derivatives can play a role in the computation of an exact Hessian of Ψ or Φ .

The REML covariance matrix estimate minimizes the REML objective function Ψ under the constraints (7). When the objective function Ψ is bounded from below, REML estimation can be seen either as the minimization of Ψ or as the solution of the system of non-linear equations $\nabla\Psi = 0$. These two approaches are not mathematically equivalent when Ψ is unbounded. The default PETSc unconstrained minimization algorithm with line search is used here for the sake of demonstrating the scalability of the proposed gradient algorithms, see Moré and Thunent [33] for details. It should be noted that the solution of the corresponding system of non-linear equations, also implemented in PETSc, requires no computation of the likelihood objective function and might be used with a parameterization $C = W^T W$ where W is not triangular.

Newton-Raphson based methods are iterative methods that solve exactly a quadratic approximation of the original optimization problem near the current approximation for the optimal Θ^* . These methods compute at each step the REML gradient vector and either an exact or an approximate Hessian matrix. Not all optimization techniques require the use of a Hessian matrix or even of a gradient vector. For instance, Conjugate Gradient methods do not require exact Hessian matrices and are from the computational complexity point of view intermediate between steepest decent methods and Newton-Raphson, see for example [36]. These methods are not studied here but it should be noted that efficient exact gradient computations can also speedup the convergence of Conjugate Gradient methods.

A Newton-Raphson algorithm for computing the REML estimate C proceeds iteratively from some initial guess $\Theta^0 = (\theta_1^{[0]} \dots \theta_q^{[0]})$ and the corresponding $C^{[0]}$ and builds successive estimates until some convergence criterion is satisfied or lack of convergence is detected. The next REML estimate for Θ^* , say Θ^{i+1} is obtained by a translation of the current estimate Θ^i along a “most-promising” direction. This direction is determined by solving the “Hessian” system (8) or some computationally expedient approximation, where $H^{[i]}$ denotes the approximation of the Hessian of Ψ for the current estimate Θ^i and where ∇^i denotes the corresponding value of the gradient $\nabla\Psi$.

$$H^{[i]}y^i = -\nabla^i \tag{8}$$

The five basic components of a Newton-Raphson iteration are therefore the evaluation of the likelihood function Ψ , its gradient $\nabla\Psi$ and its (approximate) Hessian $H(\Psi)$, the solution of the Hessian system (8)

and the line search. This paper concentrates on the computation in parallel of the likelihood and gradient vector. The approximate Hessian \hat{H} used for this paper is the approximation that is recommended by Neumaier and Groenvelde in [34] as a good initial guess for a quasi-Newton REML algorithm. This approximate Hessian \hat{H} is defined by Equations (9) and (10) below where the column s_k ($k = 1 \dots q$) of the $m \times q$ matrix S is the vector $(\partial C / \partial \theta_k)u$. It is easy to see how \hat{H} can be computed by first computing in turn S , $C^{-1}S$ and U . The resulting operation count for the present setup is reported in Table 1 of Section 3. Exact Hessian matrices can be computed if needed by a forward differentiation of either of the gradient algorithms presented in Section 4. Such Hessian matrices are potentially much more expensive to compute than the exact gradient $\nabla \Psi$.

$$\hat{H} = S^T C^{-1} S - U^T U \quad (9)$$

$$R^T U = A^T C^{-1} S \quad (10)$$

In this paper, the Hessian system (8) is solved serially by all processes using a Krylov based method even when it is small enough that a direct solution would be more effective. Multiple stopping criteria are used in the default PETSc setting for unconstrained minimization. Convergence is detected when the objective function has decreased enough in absolute terms or relative to the initial guess, or when successive approximate solutions are close enough (with respect to the Euclidean norm). Divergence is detected when either the number of non-linear iterations or the number of function evaluations has reached some user-defined threshold.

3 Computing the likelihood

The REML objective function Ψ must be evaluated in the minimization approach to REML estimation. The determinant $|C|$ of C that appears in Equation (5) for the likelihood Ψ is readily available from the exact Cholesky factor L of C . The position of each of the parameters $\theta_1, \dots, \theta_q$ within the sparse matrix L being recorded by the matrix \tilde{L} , the number η_k of occurrences of each θ_k along the main diagonal of L can be computed once at the beginning. These η_k can be stored on all processes and used to speedup the computation of $\log|C| = 2\log|L|$ and its gradient.

The determinant of the matrix $A^T C^{-1} A$ can be computed from a QR factorization of this matrix. Pan, Yu and Stewart in [38] have shown that a QR factorization leads to a more accurate determinant computation than a LU factorization does. Since in the present case, the exact Cholesky factor L of C is known, the determinant of $A^T C^{-1} A$ can be computed from a skinny QR factorization of the matrix $B = L^{-1} A = QR$. Here, the columns of the matrix B are computed by solving n independent $m \times m$ lower triangular systems of linear equations. The latter parallelism is not exploited for this paper. Note that higher than default accuracy is needed from the iterative solvers in order for the relation $A = LQR$ to hold numerically. The skinny QR factorization of B can be computed by a distributed modified Gram-Schmidt (MGS) algorithm. Figure 1 shows pseudocode for the corresponding computation of the logarithm ζ of the determinant of $B^T B = R^T R$. When the matrix B is sparse, the determinant of $B^T B$ can be computed from either its Cholesky factorization or a sparse QR factorization of B , see for example [35]. An interesting point to note is that the augmented system (7) is well known from the solution of the discretized Navier Equations. Thus, the vectors u and x can be computed directly from (7) using a variety of iterative algorithms when A is nearly column rank deficient.

The residual term $r^T C^{-1} r$ that appears in the objective function Ψ is equal to $u^T C u$. The vector u is central to REML estimation, it can be computed by solving the augmented system (7). Alternatively, one can employ a skinny QR factorization of B to express u independently of x as shown below in Equation (11).

$$u = -L^{-T} (I - QQ^T) L^{-1} b \quad (11)$$

In the present context, the matrices Q and R are computed anew each time the vector Θ is updated. The matrix Q being dense, the right hand side of Equation (11) can be computed efficiently. When the matrix B is large and sparse, it may have to be reordered to produce a sparse matrix Q . Alternatively, one could in theory use the matrix Q implicitly since multiplication by $I - QQ^T$ computes an orthogonal projection onto the null space of $B = L^{-1} A$. Such an approach may be practical on a parallel computer

```

do  $i = 1, n$ 
   $Q_i^0 = L^{-1}A_i$ 
  do  $j = 1, i - 1$ 
     $Q_i^j = Q_i^{j-1} - (Q_j^T Q_i^{j-1}) Q_j$ 
  end do
   $\mu_i = (Q_i^{i-1})^T (Q_i^{i-1})$ 
   $Q_i^i = (Q_i^{i-1}) / \mu_i^{\frac{1}{2}}$ 
end do
 $\zeta = \sum_{i=1}^n 2 \log \mu_i$ 

```

Figure 1: Computing $\zeta = \log |A^T C^{-1} A|$ from a modified Gram-Schmidt Factorization of $B = L^{-1} A$

	$K^{-1}z$		Kz		$y^T z$		$\hat{K}z$
	$\Re^{m \times m}$	$\Re^{n \times n}$	$\Re^{m \times m}$	$\Re^{m \times n}$	\Re^m	\Re^n	$\Re^{m \times m}$
Ψ	$n + 2$	0	0	1	$\frac{n^2 + 3n}{2}$	0	0
\hat{H}	$2q$	q	q	q	q^2	q^2	$2q$

Table 1: Major computational kernels used in the evaluation of the REML objective function Ψ and the corresponding approximate Hessian.

when a good sparse approximate inverse of L is available, see for example [20]. Note also that each column of the Jacobian of u with respect to Θ can be defined in terms of the solution of a system of linear equations with the same coefficient matrix as (7). The initial cost of an approximate sparse inverse of the augmented matrix in (7) may be justified in this context.

To conclude this section, Table 1 shows how many solutions of linear system ($K^{-1}z$), additional matrix-vector multiplications (Kz) and inner products ($y^T z$) are employed to evaluate the objective function Ψ and the corresponding approximate Hessian \hat{H} . The vector scatter operation, labeled ($\hat{K}z$) in Table 1 denotes a redistribution of one length m distributed vector. The vector scatter operations recorded for the evaluation of Ψ are due to the computation of the transpose L^T of L . The latter matrix is needed by some of the iterative solvers in PETSc. Since each row of the matrix L is local to one process, the transpose L^T can be computed by scattering each row of L onto all processes. Please note that throughout this paper, operation counts **and runtimes** reported under the column headings $K^{-1}z$, Kz and $\hat{K}z$ always take into account operations with both the matrix K and its transpose if any. Linked triads $y = \alpha x + y$ of distributed vectors, also known as saxpys, and multiplication of distributed vectors by a scalar are not accounted for in Table 1. Typically, these operations do not have an adverse impact on the scalability of the overall algorithm.

The likelihood evaluation algorithm described here allocates two $m \times n$ distributed dense matrices, namely B and its orthogonal factor Q . The $n \times n$ triangular factor R of B is replicated onto all processes.

4 Gradient Computations

Much floating-point arithmetic can be avoided during the computation of the REML gradient when the sparsity pattern of the covariance matrix is known to be block diagonal, say with $C = \text{diag}(C_1, \dots, C_r)$. In that case, Neumaier and Groenveld recommend using the following equations, once translated in the notation of the present paper.

$$P_\epsilon = C^{-1} - C^{-1} A (B B^T)^{-1} A^T C^{-1} - u u^T \quad (12)$$

$$\frac{\partial \Psi}{\partial \theta_k} = \text{trace} \left(P_\epsilon \frac{\partial C}{\partial \theta_k} \right) \quad (13)$$

$$= \text{vec} (P_\epsilon)^T \text{vec} \left(\frac{\partial C}{\partial \theta_k} \right) \quad (14)$$

The i^{th} diagonal block of $P_\epsilon + uu^T$ is $C_i^{-1}A_i(BB^T)^{-1}A_i^T C_i^{-1}$ where A_i is the block of rows of A corresponding to the i^{th} diagonal block C_i of C . Equation (14) then provides an efficient way of computing the gradient $\nabla\Psi$. Neumaier and Groenveld use this expression in the context of a sparse exogenous matrix A . They replace the exact inverse of BB^T by a sparse approximate inverse $(BB^T)^-$ having the same sparsity structure as BB^T in order to reduce further the cost of computing $\nabla\Psi$. A drawback of Equation (14) when C has a general sparsity pattern is that C^{-1} may be dense. A gradient computation based on Equation (14) then involves computing a dense $m \times m$ matrix and may become prohibitive when m is large. This problem can be addressed in several ways, two of which are described in the next subsections.

First, the gradient $\nabla\Psi$ can be decomposed as the sum of three independent gradients corresponding to the three terms on the right hand side of Equation (5). The gradient of the residual term $u^T C u$ can be computed by using the product rule of derivatives and by observing that $(\partial r / \partial \theta_k) = -A(\partial x / \partial \theta_k)$. The resulting expression shown below can be computed in one vector-scatter, one matrix-vector multiplication and one inner product. The matrix-vector multiplication on the right of the second equation is the same for all distribution parameters and need not be repeated.

$$\begin{aligned} \frac{\partial r^T C^{-1} r}{\partial \theta_k} &= -u^T \frac{\partial C}{\partial \theta_k} u \\ &= -2 \left(\frac{\partial L^T}{\partial \theta_k} u \right)^T (L^T u) \end{aligned}$$

The gradient of $\log(|C|)$ is readily computed from the exact Cholesky factor L of C . Recall that η_k denotes the number of occurrences of each θ_k along the main diagonal of L . With this notation, the k^{th} element of the gradient $\nabla \log(|C|)$ is shown in Equation (15) below.

$$\frac{\partial \log |C|}{\partial \theta_k} = 2\eta_k \theta_k^{-1} \quad (15)$$

The gradient of the logarithm ζ of the determinant $|B^T B| = |R|^2$ is the numerically intensive part of the computation of $\nabla\Psi$. This gradient can be computed by using a mixture of techniques for structured adjoint and tangent derivations [47]. The same gradient can also be computed from Cramer's formula together with structured tangent derivations. The corresponding two gradient algorithms are presented next.

4.1 Gradients based on structured adjoint computation

The computation of ζ in Figure 1 above can be seen as a composition of three operations, namely the solution of $LB = A$, the QR factorization $QR = B$ and a reduction operation to compute ζ from $\mu_1 = R_{1,1}^2, \dots, \mu_n = R_{n,n}^2$. The gradient of each of these operations can be combined using the chain rule. In the forward mode of differentiation, each assignment in Figure 1 is annotated with other assignments that compute the partial derivative of the corresponding left hand side variable with respect to one of the θ_k . For example, the assignment to Q_i^0 in Figure 1 would be matched by the assignment shown below in Equation (16).

$$\dot{Q}_i^0 = -L^{-1} \dot{L} A_i \quad (16)$$

Figure 2 shows how the pseudocode in Figure 1 can be augmented to compute the derivatives \dot{Q} , \dot{R} and $\dot{\zeta}$ with respect to θ_k of Q , R and ζ respectively. The superscripts to the columns of the matrix Q have been omitted for the sake of conciseness. This differentiation process is repeated for each element of the vector Θ . The amount of arithmetic needed to compute $\nabla\zeta$ in the forward mode is therefore q times that of computing ζ in the first place. In practice, some inner products need not be repeated because the matrix R is available to all processes.

```

Begin Forward( $\dot{B}, \dot{Q}, \dot{R}, \zeta, \dot{\zeta}$ )
 $d = 0.0$  ;  $\dot{d} = 0.0$ 
 $Q = L^{-1}A$  ;  $\dot{Q} = -L^{-1}\dot{L}A$ 
do  $h = 1, n$ 
  doj = 1,  $h - 1$ 
     $r_{jh} = Q_j^T Q_h$  ;  $\dot{r}_{jh} = Q_j^T \dot{Q}_h + \dot{Q}_j^T Q_h$ 
     $Q_h = Q_h - r_{jh} Q_j$  ;  $\dot{Q}_h = \dot{Q}_h - \dot{r}_{jh} Q_j - r_{jh} \dot{Q}_j$ 
  end do
   $r_{hh} = (Q_h^T Q_h)^{\frac{1}{2}}$  ;  $\dot{r}_{hh} = Q_h^T \dot{Q}_h / r_{hh}$ 
   $d = d + 2 * \log(r_{hh})$  ;  $\dot{d} = \dot{d} + 2 * \dot{r}_{hh} r_{hh}^{-1}$ 
   $Q_h = Q_h / r_{hh}$  ;  $\dot{Q}_h = \dot{Q}_h r_{hh}^{-1} - \dot{r}_{hh} Q_h r_{hh}^{-2}$ 
end do
End Forward

```

Figure 2: Forward mode computation of the tangent of a modified Gram-Schmidt factorization with respect to one model parameter θ_k .

assignment	adjoint
$c = a - (a^T b)b$	$\bar{a} = \bar{c} - (\bar{c}b) b$ $\bar{b} = - (a^T b) \bar{c} - (\bar{c}b) a$
$\beta = a^T a$	$\bar{a} = 2\beta a$
$b = \beta^{-\frac{1}{2}} a$	$\bar{\beta} = -\frac{1}{2} \bar{b} a \beta^{-\frac{3}{2}}$ $\bar{a} = \beta^{-\frac{1}{2}} \bar{b}$

Table 2: Annotations used for a reverse mode differentiation of some of the main statements in a QR factorization

In the reverse (or backward) mode of differentiation, the scalar ζ is computed once. The adjoints $\bar{\iota} = (\partial\zeta/\partial\iota)^T$ of each intermediate value ι with respect to ζ are then computed starting from $\iota = \zeta$ until $\iota = \Theta$ which yields all the desired partial derivatives $\partial\zeta/\partial\theta_k$. Note that when ι denotes a matrix: $(\bar{\iota}_{ij}) = (\bar{\iota})_{ji}$. Adjoints are propagated using the chain rule. The adjoint code for the main statements of the QR factorization of B are shown in Table 2 where a , b and c are any conformal vectors and β is any positive scalar. The amount of floating point arithmetic required by a reverse mode differentiation is a constant multiple of the amount of floating point arithmetic needed by the original computation. However, all intermediate values must be available at the proper time in order for adjoints to propagate. The adjoints of μ_1, \dots, μ_q with respect to ζ are readily computed in reverse mode since their contributions to ζ can be summed independently. The intermediate vectors Q^j used in Figure 1 to compute the QR factorization of the dense matrix B can be recomputed during the reverse phase from the matrices Q and R and need not be stored. This reconstruction requires only vector saxpys, of the form $y = \alpha x + y$, since the matrix R is replicated onto every process. Note that if the matrix R is not saved, it can be recomputed from the Q_j 's and the μ_j 's with one inner product on average per element of the upper triangular part of R . The adjoint of L with respect to ζ may be computed in reverse mode once the adjoint of B has been computed. However, a naive implementation of this approach would result in m matrix-vector multiplications and, for large enough m , may be prohibitive on multicomputers with a large enough latency. This is because any rank-1 update $E = F - yz^T$ taking place in a triangular solve with $E, F \in \mathbb{R}^{r \times s}$, $y \in \mathbb{R}^r$ and $z \in \mathbb{R}^s$ is augmented in the reverse mode by the three assignments as shown below where $\bar{E}, \bar{F} \in \mathbb{R}^{s \times r}$, $\bar{y}^T \in \mathbb{R}^r$ and $\bar{z}^T \in \mathbb{R}^s$.

$$\begin{aligned} \bar{F} &= \bar{F} + \bar{E} \\ \bar{y} &= \bar{y} - z^T \bar{E} \end{aligned}$$

```

 $\overline{Q} = 0$ 
do  $k = 1, n$ 
   $\overline{\mu}_k = r_{kk}^{-2}$ 
end do
do  $k = n, 1, -1$ 
   $Q_k = r_{kk} Q_k$ 
   $\overline{\mu}_k = \overline{\mu}_k - \frac{1}{2} \overline{Q}_k Q_k r_{kk}^{-3}$ 
   $\overline{Q}_k = \overline{Q}_k / r_{kk} + 2 \overline{\mu}_k Q_k^T$ 
  do  $j = k - 1, 1, -1$ 
     $Q_k = Q_k + r_{jk} Q_j$ 
     $w = \overline{Q}_k Q_j$ 
     $\overline{Q}_j = \overline{Q}_j - r_{jk} \overline{Q}_k - w Q_k^T$ 
     $\overline{Q}_k = \overline{Q}_k - w Q_k^T$ 
  end do
end do
do  $k = 1, q$ 
   $\nabla \zeta_k = - \sum_{j=1}^n \overline{Q}_j L^{-1} \frac{\partial L}{\partial \theta_k} B_j$ 
end do

```

Figure 3: Computing the gradient of $\zeta = \log|A^T C^{-1} A|$ using a combination of forward and backward mode differentiation.

$$\overline{z} = \overline{z} - y \overline{E}^T$$

In the present case, the matrices E and F would be dense while the vectors v and w would be sparse. Such matrix-vector operations are not supported by the current sparse BLAS standard. In place of a fully adjoint computation of the gradient of ζ , the partial derivatives of B with respect to the elements of Θ can be combined with the adjoint of B using the Chain Rule. These partial derivatives require q scatters of a length m vector and qn linear solves with L as coefficient matrix. This is the approach taken for the present paper and the corresponding pseudocode is shown in Figure 3.

4.2 Gradients based on Cramer's Formula

The above gradient algorithm requires the solution of qn linear systems and many distributed inner products from tangent computations. Another gradient algorithm can be derived from Cramer's Formula (17) where W is any square matrix.

$$\frac{\partial \log|W|}{\partial \theta_k} = \text{vec} (W^{-T})^T \text{vec} \left(\frac{\partial W}{\partial \theta_k} \right) \quad (17)$$

It follows from Equation (17) that each component of $\nabla \zeta$ can be computed according to Equation (18) below. This formulation has the advantage that the same systems of linear equations are solved for all the elements of Θ . In the present context, the columns of the inverse of the matrix $B^T B$ are computed serially since the matrix R fits in the local memory of each process. The resulting algorithm is shown in Figure 4. Note that if $B^T B$ is large and sparse its inverse may also be replaced by a sparse approximate inverse.

$$\frac{\partial \zeta}{\partial \theta_k} = -2 \text{vec} \left([B^T B]^{-1} \right)^T \text{vec} \left(B^T L^{-1} \frac{\partial L}{\partial \theta_k} B \right) \quad (18)$$

Table 3 shows in conclusion the cost of evaluating the gradient $\nabla \Psi$ using a forward mode computation, the mixed structured adjoints and tangent approach of 4.1 and Cramer's formula as in 4.2 once the likelihood has been computed. The column headings are explained in Section 3 with respect to Table 1. All three gradient algorithms allocate an additional $m \times n$ distributed dense matrix to store either gradient or adjoint values.

```

solve  $L^T F = B$ 
set  $U = (B^T B)^{-1}$ 
do  $k = 1, q$ 
   $w = 0.0$ 
  do  $j = 1, n$ 
     $w = w + U_j^T F^T (\partial L / \partial \theta_k) B_j$ 
  end do
   $\nabla \kappa_k = w$ 
end do

```

Figure 4: Computation of the gradient of $\kappa = \log |A^T C^{-1} A|$ using Cramer’s formula

Method	$K^{-1}z$		Kz		$y^T z$		$\tilde{L}z$
	$\mathbb{R}^{m \times m}$	$\mathbb{R}^{n \times n}$	$\mathbb{R}^{m \times m}$	$\mathbb{R}^{m \times n}$	\mathbb{R}^m	\mathbb{R}^n	$\mathbb{R}^{m \times m}$
Forward	$qn + n$	0	0	0	$qn^2 + q$	0	q
Mixed	qn	0	0	0	$\frac{n^2+n}{2} + qn + q$	0	$qn + q$
Cramer	n	n	0	qn	q	qn	$qn + q$

Table 3: Major computational kernels of evaluating the gradient of the REML objective function Ψ .

5 Numerical experiments and parallel performance

Thorough experimentation of the computational complexity of parallel numerical algorithms can only be based on problems whose sizes can be varied in arbitrary ways, which is seldom the case for data sets from real life applications. It is on the other hand much less easy to generate synthetic REML problems with a known solution vector Θ^* than it is for the problem of solving systems of equations. A trade-off is to generate, for some given m , n and sparsity pattern \tilde{L} , a problem with a solution that is likely nearby a known point. Precisely, a random matrix A is computed whose entries are independent and uniformly distributed between 0 and 1. A vector v is also computed whose components are independent but normally distributed with mean 0 and variance 1. Once a matrix L has been defined for some choice of Θ , the response vector b is defined as $b = A1 + Lv$ where 1 is a vector of units. There is no guarantee that the REML covariance matrix estimate found for such synthetic problems will be $L * L^T$, moreover the estimate computed by Newton-Raphson need not be globally optimum. The REML algorithms described above were validated by comparing their rates of decrease of the objective function and of the norm of the gradient. The three gradient algorithms of Section 4 were found to be interchangeable, in the sense that they produced the same sequence of objective function values and converged to similar distribution and model parameters. The approximate Hessian algorithm was verified using an exact Hessian algorithm that consists in a forward differentiation of the mixed adjoint-tangent gradient algorithm of Subsection 4.1.

For the first series of test cases, the matrix L is block diagonal with a single lower triangular block being repeated. Each non-zero element of this diagonal block is a distinct element of the vector Θ . The linear system solver is the default PETSc solver, i.e. restarted GMRES. Table 4 shows the impact of varying the number m of equations in the linear model on the runtime of REML estimation with gradients evaluated with the mixed mode and Cramer algorithms. The linear models all have 10 model parameters, 10 distribution parameters and a block diagonal covariance matrix. Most of the time spent evaluating Ψ is spent computing the matrix B , its QR factorization and the vector $L^T u$. All these quantities are used in the subsequent gradient or Hessian computation. The approximate Hessian \hat{H} remains inexpensive to compute as the number m of equations is increased. The overall runtimes, labeled REML, are roughly linear in the number of equations. Table 5 illustrates the impact of additional model parameters on REML runtime using either the mixed adjoint-tangent or the Cramer based gradient algorithms. The latter approach is clearly more efficient. Table 5 shows that for large enough values of n the computation of B and its QR factorization become dominant. This is in accordance with the operation counts in Tables 1 and 3. The skinny QR factorization of B used here is based on level-1 BLAS operations.

secs m	Mixed Mode				Cramer			
	REML	Ψ	$\nabla\Psi$	\hat{H}	REML	Ψ	$\nabla\Psi$	\hat{H}
10000	86.8	13.1	70.4	3.3	56.1	13.7	39.1	3.3
50000	124.0	17.0	102.1	4.9	64.5	17.0	42.5	5.0
100000	173.6	23.4	143.6	6.6	78.6	22.0	50.2	6.4
150000	205.8	27.3	170.2	8.3	92.7	27.1	57.3	8.3

Table 4: Run time on 32 processors of an IBM SP of the first five REML iterations as a function of the number m of equations. The covariance matrix C is block diagonal with identical 4×4 blocks and there are 50 model parameters.

secs n	Mixed Mode				Cramer			
	REML	Ψ	$\nabla\Psi$	\hat{H}	REML	Ψ	$\nabla\Psi$	\hat{H}
20	25.4	3.4	19.1	3.0	22.1	3.4	15.6	3.1
40	61.3	9.2	40.0	3.1	42.3	9.4	29.8	3.1
60	110.2	16.9	89.8	3.5	66.1	17.4	45.2	3.5
80	172.9	27.8	141.3	3.7	92.4	27.9	60.8	3.7
100	251.0	40.9	205.6	4.6	123.7	41.7	77.6	4.4

Table 5: Run time on 32 processors of an IBM SP of the first five REML iterations as a function of the number n of model parameters. The covariance matrix C is block diagonal with identical 4×4 blocks and there are 10,000 equations.

Efficient implementations of MGS based on level-3 BLAS and collective communications could be used instead, see for example [24]. A fully adjoint based computation of $\nabla\Psi$ would require fewer solutions of $m \times m$ linear systems than the mixed mode computation of $\nabla\Psi$ and may outperform the Cramer-based algorithm when the block size is large enough. As discussed in Section 4 such a gradient algorithm would might rely on parallel sparse BLAS subroutines not included in the current Sparse BLAS standard.

Table 6 illustrates the impact of the number of distribution parameters on the REML runtime for block diagonal linear models with 50,000 equations and 50 model parameters. The runtime is roughly linear in the number of distribution parameters. The remaining tables illustrate the scalability of REML estimation based on Cramer’s Formula. Table 7 shows the runtime of the first three Newton-Raphson iterations for a linear model with using Cramer’s Formula on an IBM SP and a Cray T3E. The linear model has a block diagonal covariance matrix with one million equations and 10×10 diagonal blocks, which yields 55 distribution parameters. There are 10 model parameters. As the number of processors is increased, the costs of inner products and vector scatter operations increase to the point where there is no speedup. This example shows the importance of a small inter-processor latency.

In the second series of test problems, the matrix L has q consecutive bands below the diagonal with a distinct element of Θ along each band. The inverse of the resulting matrix L is typically dense. Traditional

secs q	Mixed Mode				Cramer			
	REML	Ψ	$\nabla\Psi$	\hat{H}	REML	Ψ	$\nabla\Psi$	\hat{H}
1	77.8	15.5	61.6	0.7	24.3	16.0	7.6	0.7
3	90.8	16.8	72.3	1.7	35.5	17.1	16.8	1.6
6	106.2	18.6	84.7	2.9	48.3	17.2	28.2	2.9
10	122.2	17.1	100.2	4.9	68.6	18.3	45.6	4.7
15	149.6	17.4	125.2	7.0	86.3	17.2	62.2	6.8

Table 6: Run time on 32 processors of an IBM SP of five REML iterations as a function of the number q of distribution parameters. The covariance matrix C is block diagonal with 50,000 rows. There are 50 model parameters.

secs	IBM SP			CRAY T3E		
	Iteration			Iteration		
p	1	2	3	1	2	3
16	66.2	130.5	129.4			
32	38.6	73.3	73.4	13.2	27.3	28.7
64	30.9	56.0	53.7	8.5	16.3	16.4
128	34.2	53.4	54.1	6.7	11.2	11.1
256				7.7	10.5	10.4

Table 7: Run time of the first three REML iterations as a function of the number p of processes. The covariance matrix C of size 1,000,000 is block diagonal with 10x10 blocks. There are 10 model parameters.

secs	Block Jacobi				Additive Schwarz			
	REML	Ψ	$\nabla\Psi$	\hat{H}	REML	Ψ	$\nabla\Psi$	\hat{H}
16	3852	1928	1923	1.2	18.7	8.2	7.6	2.9
32	5379	2690	2689	1.7	14.2	5.7	6.2	2.3
64					14.7	5.4	7.0	2.4
128					22.8	7.9	11.0	4.0

Table 8: Scalability of REML estimation using Cramer’s Formula for a bi-diagonal factor L of size 100,000 on an IBM SP. The estimation was carried for five Newton-Raphson iterations; there are 10 model parameters.

approaches as exemplified by Equation (14) of Section 4 do not readily apply in such cases. Table 8 shows the runtime of the first 5 Newton-Raphson iterations for a bi-diagonal factor L of size 100,000. There are 10 model parameters and the gradient is computed using Cramer’s Formula. Table 8 illustrates the importance of preconditioning as the performance of block Jacobi clearly deteriorates as more processes are added. With 64 and 128 processes, there is a dramatic increase in the runtime of some of the Newton-Raphson iterations that results in a time out of the computation. The additive Schwarz preconditioner with unit overlap fares better with an optimal runtime at $p = 32$. Multigrid preconditioners are available as an advanced feature of PETSc and would likely improve the scalability of REML estimation for this test case where the covariance matrix C does not have a sparse inverse. Alternatively, since the matrix L is assumed lower triangular, it is technically possible to replace restarted GMRES by a direct triangular solve. The most effective strategy in terms of robustness and scalability is likely application specific and is not pursued further here.

So far, this paper has been concerned with REML estimation but the algorithms presented here can be specialized to compute ML estimates as well. Table 9 shows some runtime measurements on an IBM SP of ML estimation with an exact Hessian and a Cramer based gradient. The linear model is the same as for Table 7.

To summarize, two parallel algorithms for computing REML gradients have been presented and

secs	REML	Ψ	$\nabla\Psi$	\hat{H}
p				
16	5313	432.7	215.4	4666
32	2896	243.1	118.2	2535
64	1222	28.1	60.0	1136
128	831	23.6	39.0	769

Table 9: Scalability of ML estimation using Cramer’s Formula and exact Hessians for a block diagonal factor L of size 100,000 on an IBM SP. The estimation was carried for five Newton-Raphson iterations; there are 10 model parameters and 55 distribution parameters.

compared for large linear models where the inverse of the covariance matrix need not be sparse. Of these two algorithms, the one based on Cramer's Formula was shown to be faster and more scalable than one based on a mixture of adjoint and tangent computations. However, analysis of the latter algorithm suggests that an effective way to compute the REML gradient may be via full adjoint computations using on parallel BLAS routines with sparse vector arguments.

The linear system solvers and preconditioners used in this paper can all be selected by setting appropriate runtime arguments to PETSc programs. Given this constraint, the scalability of the REML estimation algorithms presented here is very good for standard shapes of covariance matrices. These REML algorithms were tested on linear models with banded covariance matrices whose inverses are dense. Such covariance matrices may for instance correspond to correlation in time. In the latter case, a reduction in runtime was observed on up to 32 processors of an IBM SP. Further speedups may be achieved via more efficient multigrid preconditioners or preconditioners in BlockSolve95 [22] and ParPre [13] or by direct sparse linear system solvers. This work shows that REML estimates of covariance matrices can be computed efficiently on distributed memory multicomputers using widely available parallel software tools.

6 Acknowledgements

I thank Jeanne Adam, Erricos Kontogiorghes, the developers of PETSC and the anonymous reviewers for their help, comments and constructive reviews.

This research was performed in part using the Molecular Science Computing Facility (MSCF) in the William R. Wiley Environmental Molecular Sciences Laboratory at the Pacific Northwest National Laboratory. The MSCF is funded by the Office of Biological and Environmental Research in the U.S. Department of Energy. Pacific Northwest is operated by Battelle for the U.S. Department of Energy under Contract DE-AC06-76RLO 1830.

This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

References

- [1] T. Aittokallio, V. Ahola, and E. Uusipaikka. Likelihood based statistical inference in hidden markov models. Technical Report TUCS Technical Report No. 260, Turku Centre for Computer Science, March 1999.
- [2] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhauser Press, 1997.
- [3] Basic Linear Algebra Subprograms Technical (BLAST) Forum. Document for the basic linear algebra subprograms (BLAS) standard. *available via the URL: <http://www.netlib.org/cgi-bin/checkout/blast/blast.pl>*, October 1999.
- [4] M. Broadley, N. Willey, and A. Mead. A method to assess taxonomic variations in shoot caesium concentration among flowering plants. *Environmental Pollution*, 106:341–9, 1999.
- [5] J. Brooke. The parallel Mln project. CS6400 User Group Meeting, San Diego, CA, November 1996.
- [6] J. M. Bull, G. D. Riley, J. Rasbash, and H. Goldstein. Parallel implementation of a multilevel modelling package. *Computational Statistics and Data Analysis*, 31(4):457–474, 1999.
- [7] J. Carrera and S. Neuman. Estimation of aquifer parameters under transient and steady state conditions. *Water Resources Research*, 22(2):199–242, 1986.
- [8] C. Ceron, J. Dopazo, E. Zapata, J. Carazo, and O. Trelles. Parallel implementation of DNAmI program on message-passing architectures. *Parallel Computing*, 24:701–16, 1998.

- [9] R. Clarke. Residual maximum likelihood (reml) methods for analysing hydrological data series. *Journal of Hydrology*, 182:277–95, 1996.
- [10] A. Conn, N. Gould, D. Orban, and P. Toint. A primal-dual trust-region algorithm for minimizing a non-convex function subject to general inequality and linear equality constraints. Technical Report RAL-TR-1999-054, Rutherford Appleton Laboratory, Oxfordshire, UK, 1999.
- [11] J. Dongarra, I. Duff, D. Sorensen, and H. van der Vorst. *Numerical Linear Algebra for High-Performance Computers*. SIAM Press, Philadelphia, 1998.
- [12] I. Duff, A. Erisman, and J. Reid. *Direct Methods for Sparse Matrices*. Monographs on Numerical Analysis. Clarendon Press, Oxford, 1986.
- [13] V. Eijkhout and T. Chan. ParPre: a parallel preconditioners package; reference manual for version 2.0.17. Technical Report CAM report 97-24, University of California, Los Angeles, 1997.
- [14] D. Falconer and T. Mackay. *Introduction to Quantitative Genetics, fourth edition*. Longman, Essex UK, 1996.
- [15] W. Fellner. Sparse matrices, and the estimation of variance components by likelihood methods. *Communications in Statistics - Part B: Simulation and Computation*, 16(2):439–63, 1987.
- [16] C. Fraley and P. Burns. Large-scale estimation of variance and covariance components. *SIAM Journal on Scientific Computation*, 16(1):192–209, January 1995.
- [17] G. Golub and C. van Loan. *Matrix Computations*. John Hopkins University Press, Baltimore, 1989.
- [18] Gouriéroux, Monfort, and Renault. Indirect inference. *Journal of Applied Econometrics*, 8:S85–118, 1993.
- [19] D. Harville. Maximum likelihood approaches to variance components and related problems. *Journal of American Statistical Association*, 72:320–40, 1977.
- [20] T. Huckle and M. Grote. A new approach to parallel preconditioning with sparse approximate inverses. Technical Report SCCM-94-03, Stanford University, Stanford, CA, 1994.
- [21] H. Jones, G. Mitra, D. Parkinson, and T. Spinks. A parallel implementation of the maximum likelihood method in positron emission tomography image reconstruction. *Computational Statistics and Data Analysis*, 31:417–39, 1999.
- [22] M. Jones and P. P.E. Blocksolve95 users manual: Scalable library software for the parallel solution of sparse linear systems. Technical Report ANL-95/48, Argonne National Laboratory, December 1995.
- [23] E. Kontoghiorghes. *Parallel Algorithms for Linear Models*. Kluwer Press, Dordrecht, Netherlands, 2000.
- [24] J. Malard and C. Paige. Efficiency and scalability of two parallel QR factorization algorithms. In *Scalable High Performance Computation Conference 94*, pages 615–22, Knoxville, TN, May 1994.
- [25] P. Matstoms. Sparse QR factorization in MATLAB. *ACM Transactions on Mathematical Software*, 20(1):136–159, Mar. 1994.
- [26] H. Matsuda, G. Olsen, R. Overbeek, and Y. Kaneda. Fast phylogenetic analysis on a massively parallel machine. In *Proceedings of the International Conference on Supercomputing, Manchester UK*, July 1994.
- [27] G. McLachlan and K. Thriyambakam. *The EM Algorithm and Extensions*. Probability and Statistics. John Wiley & Sons, New York, 1996.
- [28] W. Meiring, P. Guttorp, and P. Sampson. Space-time estimation of grid-cell hourly ozone levels for assessment of a deterministic model. *Environmental & Ecological Statistics*, 5:197–222, 1998.

- [29] K. Meyer. Estimating variances and covariances for multivariate animal models by restricted maximum likelihood. *Genet. Sel. Evol.*, 23:67–83, 1991.
- [30] I. Misztal. Restricted maximum likelihood estimation of variance components in animal model using sparse matrix inversion and a supercomputer. *Journal of Dairy Science*, 73:163–72, 1990.
- [31] I. Misztal. A comparison of computing properties of derivative and derivative-free algorithms in variance-component estimation by REML. *Journal Animal Breeding*, 111:346–55, 1994.
- [32] R. Möller. A systolic implementation of the MLEM reconstruction algorithm for positron emission tomography images. *Parallel Computing*, 25:905–20, 1999.
- [33] J. J. Moré and D. J. Thuente. Line search algorithms with guaranteed sufficient decrease. *ACM Transactions on Mathematical Software*, 20(3):286–307, 1994.
- [34] A. Neumaier and E. Groeneveld. Restricted maximum likelihood estimation of covariances in sparse linear models. *Genet. Sel. Evol.*, 30:3–26, 1998.
- [35] E. Ng and P. B.W. Some results on structure prediction in sparse qr factorization. *SIAM J. on Matrix Analysis and Applications*, 17(2), 1996.
- [36] L. Ortiz and L. Kaelbling. Notes on methods based on maximum-likelihood estimation for learning the parameters of the mixture of gaussian models. Technical Report CS-99-03, Department of Computer Science, Brown University, 1999.
- [37] C. Paige. Fast numerically stable computations for generalized linear least squares problems. *SIAM J. Num. Anal.*, 16:165–71, 1979.
- [38] V. Y. Pan, Y. Yu, and C. Stewart. Algebraic and numerical techniques for the computation of matrix determinants. *Computers Math. Applic.*, 34(1):43–70, 1997.
- [39] H. Patterson and R. Thompson. Recovery of interblock information when block sizes are unequal. *Biometrika*, 58:545–54, 1971.
- [40] G. Pflug and Świątanowski. Selected parallel optimization methods for financial management under uncertainty. *Parallel Computing*, 26:3–25, 2000.
- [41] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Computer Science/Numerical Methods. PWS Publishing Co., Boston, MA, 1996.
- [42] R. Schnabel. A view of the limitations, opportunities, and challenges in parallel nonlinear optimization. *Parallel Computing*, 21:875–905, 1995.
- [43] G. Smith and G. M.J. Models of wheat grain quality considering climate, cultivar and nitrogen effects. *Agricultural and Forest Meteorology*, 94:159–70, 1999.
- [44] R. Thompson. The estimation of heritability with unbalanced data i. observations available on parents and offspring. *Biometric*, 33:485–95, September 1977.
- [45] R. Thompson. The estimation of heritability with unbalanced data ii. data available on more than two generations. *Biometric*, 33:497–504, September 1977.
- [46] R. van der Oost, E. Vindimian, P. van den Brink, K. Satumalay, H. Heida, and N. Vermeulen. Biomonitoring aquatic pollution with feral eel (*anguilla anguilla*). iii. statistical analyses of relationships between contaminant exposure and biomarkers. *Aquatic Toxicology*, 39:45–75, 1997.
- [47] A. Verma. *Structured Automatic Differentiation*. PhD thesis, Department of Computer Science, Cornell University, 1998.
- [48] D. Zimmerman and M. Zimmerman. A comparison of spatial semivariogram estimators and corresponding ordinary Kriging predictors. *Technometrics*, 33:77–91, 1991.