MATHEMATICS OF COMPUTATION
Volume 80, Number 275, July 2011, Pages 1585–1600
S 0025-5718(2011)02451-8
Article electronically published on January 6, 2011

DIFFERENTIATION OF MATRIX FUNCTIONALS USING TRIANGULAR FACTORIZATION

F. R. DE HOOG, R. S. ANDERSSEN, AND M. A. LUKAS

ABSTRACT. In various applications, it is necessary to differentiate a matrix functional $w(\mathbf{A}(\mathbf{x}))$, where $\mathbf{A}(\mathbf{x})$ is a matrix depending on a parameter vector $\mathbf{x}$. Usually, the functional itself can be readily computed from a triangular factorization of $\mathbf{A}(\mathbf{x})$. This paper develops several methods that also use the triangular factorization to efficiently evaluate the first and second derivatives of the functional. Both the full and sparse matrix situations are considered. There are similarities between these methods and algorithmic differentiation. However, the methodology developed here is explicit, leading to new algorithms. It is shown how the methods apply to several applications where the functional is a log determinant, including spline smoothing, covariance selection and restricted maximum likelihood.

1. Introduction

This paper is concerned with the evaluation of functionals and their derivatives of a matrix $\mathbf{A}(\mathbf{x}) \in \mathbb{R}^{n\times n}$, which depends on a vector of parameters $\mathbf{x} \in \mathbb{R}^p$. The original motivation for this study was to find an efficient algorithm for calculating the generalized cross-validation (GCV) and robust generalized cross-validation (RGCV) scores for smoothing splines. For GCV [11, 23], it is necessary to calculate the trace of the smoothing matrix, $\mathbf{S}(x)$ say, while, for RGCV [14], it is also necessary to calculate the trace of $\mathbf{S}^2(x)$. In this situation, the vector of parameters $\mathbf{x}$ becomes the scalar smoothing parameter $x > 0$. For polynomial smoothing splines of degree $2m-1$, with $n+m$ points, $\mathbf{S}(x)$ has the form [14]

$$\mathbf{S}(x) = \mathbf{I} - x\mathbf{C}\mathbf{A}^{-1}(x)\mathbf{C}^T, \tag{1.1}$$

where $\mathbf{A}(x) = \mathbf{B} + x\mathbf{C}^T\mathbf{C}$ is a matrix pencil, $\mathbf{B}$ is a symmetric, positive definite banded matrix with bandwidth $2m-1$ and $\mathbf{C}$ is an $(n+m)\times n$, banded lower triangular matrix with bandwidth $m+1$. A direct evaluation of the quantities $\operatorname{tr}(\mathbf{S}(x))$ and $\operatorname{tr}(\mathbf{S}^2(x))$ is expensive for large n. It is easy to verify (cf. [9]) that

$$\operatorname{tr}(\mathbf{C}\mathbf{A}^{-1}\mathbf{C}^T) = \frac{d}{dx}\log\det\mathbf{A}(x), \tag{1.2a}$$

$$\operatorname{tr}(\mathbf{C}\mathbf{A}^{-1}\mathbf{C}^T)^2 = -\frac{d^2}{dx^2}\log\det\mathbf{A}(x), \tag{1.2b}$$

Received by the editor May 5, 2009.
2010 *Mathematics Subject Classification.* Primary 15A24; Secondary 15A15, 40C05, 65F30.
Key words and phrases. Differentiation matrix functionals, triangular factorization, algorithmic differentiation, log-det relationships, robust generalized cross validation, smoothing splines, REML, covariance selection.

from which the required quantities can easily be derived. The advantage of formulating the problem as one of differentiating a matrix functional, $\log\det \mathbf{A}(x)$ in this case, is that derivatives can generally be calculated with about the same complexity as the functional itself [8]. Because the matrix $\mathbf{A}(x)$ is symmetric, positive definite and banded with bandwidth $2m+1$, the matrix functional can be calculated in $O(m^2n)$ operations using a Cholesky decomposition. The application of the results in this paper to the evaluation of the quantities on the right-hand side of (1.1) yields two $O(m^2n)$ algorithms, one of which is new, for calculating the GCV score, and two new $O(m^2n)$ algorithms for calculating the RGCV score for smoothing splines (Lukas et al. [15]).

It turns out that the problem of calculating derivatives of matrix functionals occurs in a number of applications, which will be described in the sequel. As in the example of smoothing splines outlined above, many of the functionals involve determinants or ratios of determinants, and can be readily evaluated using a triangular factorization of the corresponding matrix $\mathbf{A}(\mathbf{x})$. Other functionals can often be derived by differentiating functions of determinants. For example, if $\mathbf{A}(\mathbf{x})$ is an information matrix that depends on parameters $\mathbf{x}$, then an A-optimal design, with respect to the parameters $\mathbf{x}$, minimizes $\operatorname{tr}\left(\mathbf{A}^{-1}(\mathbf{x})\right)$ [20], and this can be evaluated using [9]

$$\operatorname{tr}\left(\mathbf{A}^{-1}\right) = \frac{d}{d\varepsilon}\log\det\left(\mathbf{A}+\varepsilon\mathbf{I}\right)\bigg|_{\varepsilon=0}. \tag{1.3}$$

In fact, any functional of $\mathbf{A}^{-1}$ can be evaluated by differentiation of the log determinant using

$$\frac{\partial}{\partial a_{ij}}\log\det\mathbf{A} = \mathbf{e}_j^T\mathbf{A}^{-1}\mathbf{e}_i, \tag{1.4}$$

where $\mathbf{e}_i,\ i=1,\ \dots\ ,n$, are the usual unit vectors. However, not all functionals can be evaluated using a triangular factorization. An example is the spectral norm $\|\mathbf{A}^{-1}(\mathbf{x})\|_2$, which is minimized for an E-optimal design [20].

Different forms of the above relationships have been published covering a broad spectrum of applications [1, 7, 8, 9, 12, 19, 22]. They include a listing of properties of log determinant relationships in Harville [9], a discussion about relationships between determinants, inverses and linear systems in the context of algorithmic differentiation in Griewank [8] and Kubota [12], and efficient calculation of some elements of a matrix inverse for the analysis of short-circuit scenarios in power systems in Takahashi *et al.* [22]. For nonsparse matrices, the key relationships, at the heart of the Takahashi *et al.* [22] algorithm, can be found on page 263 of the 1945 paper of Waugh and Dwyer [24].

In a wide range of statistical applications, there is also a need to evaluate the derivative of the log determinant of some target matrix. This occurs in covariance selection in Dempster [3], high-dimensional covariance estimation by minimizing penalized log determinant divergence in Ravikumar *et al.* [19], restricted maximum likelihood (REML) calculations for mixed models in Smith [21] and statistical design calculations in Silvey [20]. The goal of this paper is to construct a rigorous and unifying computational framework in which to perform the efficient evaluation of such derivatives.

In the sequel, we investigate the evaluation of functionals and the calculation of their derivatives using triangular factorizations. Specifically, we consider a functional $w : \mathbb{R}^{n\times n} \to \mathbb{R}$, $w = w(\mathbf{A})$, and the triangular factorization

$$\mathbf{A} = \mathbf{L}\mathbf{U}, \tag{1.5}$$

where $\mathbf{L}$ is a unit lower triangular matrix and $\mathbf{U}$ is an upper triangular matrix. It is assumed that

$$g\left(\mathbf{L}, \mathbf{U}\right) := w\left(\mathbf{L}\mathbf{U}\right) = w\left(\mathbf{A}\right) \tag{1.6}$$

and its derivatives with respect to elements of $\mathbf{L}$ and $\mathbf{U}$ can be readily evaluated once the triangular factorization of $\mathbf{A}$ has been performed. Finally, we let $\mathbf{A} : \mathbb{R}^p \to \mathbb{R}^{n\times n}$, $\mathbf{A} = \mathbf{A}(\mathbf{x})$, and define the functional

$$h\left(\mathbf{x}\right) := w\left(\mathbf{A}\left(\mathbf{x}\right)\right), \tag{1.7}$$

and assume that the elements of $\mathbf{A}\left(\mathbf{x}\right)$ and their derivatives are easy to evaluate. As a consequence, the computational complexity associated with the evaluation of $h(\mathbf{x})$ will be the same as that required to determine the triangular factorization.

After some preliminaries in section 2, in section 3 we develop and analyse a direct and indirect method for calculating first derivatives of $h(\mathbf{x})$, using the triangular factorization of $\mathbf{A}(\mathbf{x})$, for both the full and sparse matrix situations. Essentially, the computational complexity of the methods is of the same order of magnitude as that required to calculate the triangular factorization (1.5). The direct and indirect methods have similarities with the forward and reverse methods, respectively, of algorithmic differentiation, but our treatment is explicit, leading to new algorithms. In section 4, we extend the methodology to the efficient calculation of second derivatives of $h(\mathbf{x})$.

In section 5, we describe several applications in which the methods of this paper can be used to perform efficient calculations. In some of the applications, the components of $\mathbf{x}$ are in fact the elements of $\mathbf{A}$ (that is, $\mathbf{x} = \text{vec}(\mathbf{A})$).

For many applications, the matrix $\mathbf{A}$ is symmetric positive definite, and the analysis in this paper can be modified, in a straightforward manner, to take advantage of symmetry by using a Cholesky factorization, rather than the *LU* factorization (1.5).

2. **Preliminaries**

In the sequel, it will be assumed that $\mathbf{A}$ is nonsingular. It is then possible to reorder the rows and/or columns of $\mathbf{A}$ so that a triangular factorization of the form (1.5) exists, and we assume that an appropriate ordering of the rows and columns has already been performed. The ordering of the rows and columns affects both the numerical stability of a triangular factorization and sparsity within the triangular components. While a detailed discussion about achieving a satisfactory ordering is beyond the scope of this paper, we note that heuristic algorithms such as the Markowitz criterion [6, 16], which chooses a pivot to minimize the fill-in by the subsequent column elimination, are still widely used. For symmetric matrices, a similar criterion leads to the widely used minimum degree algorithm [4].

For the situations where we have a sparse triangular factorization, it will be convenient to use the following notation. Let

$$\begin{aligned}\mathcal{C} &= \{(i,j)|\ l_{ij} \neq 0 \text{ or } u_{ij} \neq 0\},\\ \mathcal{L}_i &= \{k|\ k > i \text{ and } (k,i) \in \mathcal{C}\},\\ \mathcal{U}_i &= \{k|\ k > i \text{ and } (i,k) \in \mathcal{C}\},\end{aligned}$$

and

$$\mathcal{S}_{ij} = \{k|\ k \leq \min(i,j);\ (i,k) \in \mathcal{C} \text{ and } (k,j) \in \mathcal{C}\}.$$

Following standard practice in sparsity work, it is assumed that any l_{ij} or u_{ij}, that must be computed, is treated as nonzero, even if it takes a zero value due to numerical cancellation. This is formally equivalent to the assertion that [5]

$$(i,j) \in \mathcal{C} \text{ if and only if } \mathcal{S}_{ij} \neq \phi \text{ (the empty set)}. \tag{2.1}$$

Hence, if $(i,k) \in \mathcal{C}$ and $(k,j) \in \mathcal{C}$ for some $k \leq \min(i,j)$, then $(i,j) \in \mathcal{C}$.

We also use the following notation. For a matrix $\mathbf{C} \in \mathbb{R}^{n\times n}$, let $\mathbf{C}_+ \in \mathbb{R}^{n\times n}$ and $\mathbf{C}_- \in \mathbb{R}^{n\times n}$ denote the upper triangular and the strictly lower triangular part of $\mathbf{C}$, respectively. Then, for any two matrices $\mathbf{B}$ and $\mathbf{C}$ in $\mathbb{R}^{n\times n}$,

$$\operatorname{tr}\left(\mathbf{B}^T\mathbf{C}\right) = \operatorname{tr}\left(\mathbf{B}_-^T\mathbf{C}_-\right) + \operatorname{tr}\left(\mathbf{B}_+^T\mathbf{C}_+\right).$$

As discussed above, it is assumed that w and g and their derivatives can be easily evaluated. For the functional $w(\mathbf{A}) : \mathbb{R}^{n\times n} \to \mathbb{R}$, it is convenient to write its derivatives with respect to the elements of $\mathbf{A}$, as a matrix $\mathbf{W} \in \mathbb{R}^{n\times n}$ with elements given by

$$w_{ij} = \frac{\partial w(\mathbf{A})}{\partial a_{ij}},\ i,j = 1,\ldots,n. \tag{2.2}$$

We also write the derivatives of $g(\mathbf{L},\mathbf{U})$, with respect to the variable elements of $\mathbf{L}$ and $\mathbf{U}$, as a matrix $\mathbf{G} \in \mathbb{R}^{n\times n}$ with elements given by

$$g_{ij} = \begin{cases} \partial g/\partial l_{ij},\ i > j, \\ \\ \partial g/\partial u_{ij},\ i \leq j. \end{cases}$$

For a differentiable function $f : \mathbb{R}^p \to \mathbb{R}$ of a vector of parameters $\mathbf{x}$, let $f'(\mathbf{x})$ denote

$$f'(\mathbf{x}) = \mathbf{c}\cdot\nabla f = \sum_{k=1}^{p} c_k \frac{\partial f(\mathbf{x})}{\partial x_k} \tag{2.3}$$

for an arbitrary vector $\mathbf{c} \in \mathbb{R}^p$. Similarly, for a matrix $\mathbf{Z}$ with elements $z_{ij} : \mathbb{R}^p \to \mathbb{R}$, let $\mathbf{Z}'(\mathbf{x})$ denote $[z'_{ij}(\mathbf{x})] = [\mathbf{c}\cdot\nabla z_{ij}(\mathbf{x})]$. Note that if $\mathbf{c}$ is a unit vector, then $f'(\mathbf{x})$ is the usual directional derivative of f. In some applications, we are interested in the case where the parameters $\mathbf{x}$ are the matrix elements of $\mathbf{A}$ and f is a function of $\mathbf{A}$. Then we have

$$f'(\mathbf{A}) = \mathbf{c}\cdot\nabla f = \sum_{k,j=1}^{n} c_{kj}\frac{\partial f(\mathbf{A})}{\partial a_{kj}} = \left.\frac{\partial f(\mathbf{A}+\varepsilon\mathbf{C})}{\partial\varepsilon}\right|_{\varepsilon=0},$$

where $\mathbf{C} \in \mathbb{R}^{n\times n}$ denotes an arbitrary matrix and $\mathbf{c} = \operatorname{vec}(\mathbf{C})$.

3. First order differentiation

In this section, we derive and analyse procedures for direct and indirect evaluation of the first derivatives of $h(\mathbf{x})$ in (1.7).

3.1. Direct differentiation.

We have assumed that an effective way of calculating the matrix functional $w\left(\mathbf{A}\right)$ is to evaluate $g\left(\mathbf{L},\mathbf{U}\right)$. Therefore, a logical starting point for deriving an expression for derivatives of $h\left(\mathbf{x}\right)$ would appear to be the relation

$$h\left(\mathbf{x}\right) = g\left(\mathbf{L}\left(\mathbf{x}\right),\mathbf{U}\left(\mathbf{x}\right)\right), \tag{3.1}$$

where the dependence of the triangular factors $\mathbf{L}\left(\mathbf{x}\right)$ and $\mathbf{U}\left(\mathbf{x}\right)$ on the parameter $\mathbf{x}$ is defined implicitly through

$$\mathbf{L}\left(\mathbf{x}\right)\mathbf{U}\left(\mathbf{x}\right) = \mathbf{A}\left(\mathbf{x}\right). \tag{3.2}$$

On differentiating (3.1), we obtain

$$h' = \operatorname{tr}\left[(\mathbf{G}_-)^T\mathbf{L}' + (\mathbf{G}_+)^T\mathbf{U}'\right] = \operatorname{tr}\left[\mathbf{G}^T\left(\mathbf{L}' + \mathbf{U}'\right)\right], \tag{3.3}$$

which is easy to evaluate if $\mathbf{L}'$ and $\mathbf{U}'$ are known.

Differentiation of (3.2) yields

$$\mathbf{L}'\mathbf{U} + \mathbf{L}\mathbf{U}' = \mathbf{A}', \tag{3.4}$$

which provides a system of linear equations for $\mathbf{L}'$ and $\mathbf{U}'$. This system can be rewritten as

$$\left(\mathbf{L}\mathbf{U}'\right)_+ = \left(\mathbf{A}' - \mathbf{L}'\mathbf{U}\right)_+ \tag{3.5a}$$

$$\left(\mathbf{L}'\mathbf{U}\right)_- = \left(\mathbf{A}' - \mathbf{L}\mathbf{U}'\right)_-. \tag{3.5b}$$

These equations can be used to sequentially compute the rows of $\mathbf{U}'$ and the columns of $\mathbf{L}'$, starting with the first row and column, using forward substitution. As the sparsity structure of $\mathbf{U}'$ and $\mathbf{L}'$ is the same as the sparsity structure of $\mathbf{U}$ and $\mathbf{L}$, respectively, the computational effort required to calculate $\mathbf{U}'$ and $\mathbf{L}'$ is about twice the computational effort required to calculate the triangular factorization.

Since $\mathbf{G}$ is easily evaluated and $\mathbf{U}'$ and $\mathbf{L}'$ can be calculated as above, (3.3) represents a direct method for calculating the first derivative of the functional $h(\mathbf{x}) = w(\mathbf{A}\left(\mathbf{x}\right))$. In most applications, the bulk of the computational effort is associated with the calculation of $\mathbf{U}'$ and $\mathbf{L}'$. Hence, the additional computational complexity associated with direct differentiation of the functional is about twice the computational complexity associated with calculating the functional itself. In particular, if the matrix $\mathbf{A}$ is full, then the calculation of h' using (3.3) requires $O(n^2)$ operations, which is small compared to the $O(n^3)$ operations required to perform the factorization.

If, however, the matrix $\mathbf{A}$ is sparse, then the complexity of the calculation of h' using (3.3) is similar to that required to perform the factorization. In order to simplify the complexity considerations, in the sequel we will ignore the complexity associated with the final evaluation of h' using (3.3) or a similar expression.

An explicit solution for $\mathbf{U}'$ and $\mathbf{L}'$ can also be determined by pre- and post-multiplying equation (3.4) by $\mathbf{L}^{-1}$ and $\mathbf{U}^{-1}$, respectively, to obtain

$$\mathbf{L}^{-1}\mathbf{L}' + \mathbf{U}'\mathbf{U}^{-1} = \mathbf{L}^{-1}\mathbf{A}'\mathbf{U}^{-1}$$

from which it follows that

$$\mathbf{L}' = \mathbf{L}\left(\mathbf{L}^{-1}\mathbf{A}'\mathbf{U}^{-1}\right)_{-}, \qquad \mathbf{U}' = \left(\mathbf{L}^{-1}\mathbf{A}'\mathbf{U}^{-1}\right)_{+}\mathbf{U}. \tag{3.6}$$

The algorithm outlined above is closely related to the direct or forward method of algorithmic differentiation (see, for example, Griewank [8] for an overview of algorithmic differentiation or Smith [21] for its application to the Cholesky decomposition). A key difference is that the analysis above is independent of the algorithm used to obtain the triangular factorization. As a consequence, the utilization of the uncoupling of the unknowns in (3.5a) and (3.5b) (which can actually be implemented in a number of different ways) is far more transparent than would be the case for algorithmic differentiation.

Although the direct procedure outlined above provides an effective way of calculating $h'(\mathbf{x}) = \mathbf{c}\cdot\nabla h$, the whole procedure must be repeated when $\mathbf{c}$ is changed. For example, if all of the components of ∇h are required, the procedure must be repeated p times to calculate each component. Indirect differentiation, as described in the following section, addresses this issue.

3.2. **Indirect differentiation.** An alternative approach is to start with the equation

$$h\left(\mathbf{x}\right) = w\left(\mathbf{A}\left(\mathbf{x}\right)\right) \tag{3.7}$$

which can be differentiated to yield

$$h' = \operatorname{tr}\left(\mathbf{W}^T\mathbf{A}'\right). \tag{3.8}$$

At first, this does not seem particularly promising because w is difficult to evaluate directly and the elements of $\mathbf{W}$ are derivatives of w. However, it turns out that relatively simple equations can be derived for the determination of $\mathbf{W}$.

From the definition of g in (1.6),

$$w(\mathbf{LU}) = g(\mathbf{L}, \mathbf{U}).$$

Differentiation with respect to the components of $\mathbf{L}$ and $\mathbf{U}$, yields

$$\left(\mathbf{L}^T\mathbf{W}\right)_{+} = \mathbf{G}_{+}, \qquad \left(\mathbf{W}\mathbf{U}^T\right)_{-} = \mathbf{G}_{-}, \tag{3.9}$$

which can be viewed as a system of linear equations for the determination of $\mathbf{W}$. These equations can be rewritten as

$$\left(\mathbf{L}^T\mathbf{W}_{+}\right)_{+} = \mathbf{G}_{+} - \left(\mathbf{L}^T\mathbf{W}_{-}\right)_{+}, \tag{3.10a}$$

$$\left(\mathbf{W}_{-}\mathbf{U}^T\right)_{-} = \mathbf{G}_{-} - \left(\mathbf{W}_{+}\mathbf{U}^T\right)_{-}, \tag{3.10b}$$

which can be used to sequentially compute the columns of $\mathbf{W}_{+}$ and the rows of $\mathbf{W}_{-}$, starting with the last column and row, using backward substitution. The computational complexity associated with this is about twice the computational complexity associated with calculating the triangular factorization when $\mathbf{A}$ is a full matrix.

When $\mathbf{A}$ is sparse, the computational cost, relative to the triangular factorization, is substantially greater because $\mathbf{W}$ is generally a full matrix. However, on writing out (3.8) in component form

$$h' = \operatorname{tr}\left(\mathbf{W}^T\mathbf{A}'\right) = \sum_{(i,j)\in\mathcal{C}} w_{ij}a'_{ij},$$

it is clear that we only need to evaluate some of the elements of $\mathbf{W}$, namely w_{ij} with $(i,j) \in \mathcal{C}$, and it turns out that they can be calculated using a subset of the system of equations given in (3.9). Specifically, we take

$$\sum_{k=i}^{n} l_{ki}w_{kj} = w_{ij} + \sum_{k\in\mathcal{L}_i} l_{ki}w_{kj} = g_{ij},\ i \le j,\ (i,j)\in\mathcal{C}, \tag{3.11a}$$

$$\sum_{k=j}^{n} w_{ik}u_{jk} = w_{ij}u_{jj} + \sum_{k\in\mathcal{U}_j} w_{ik}u_{jk} = g_{ij},\ i > j,\ (i,j)\in\mathcal{C}. \tag{3.11b}$$

When $i \le j$ and $(i,j) \in \mathcal{C}$, it is easy to verify that $i \in \mathcal{S}_{kj}$ for all $k \in \mathcal{L}_i$, and it then follows from (2.1) that $(k,j) \in \mathcal{C}$ for all $k \in \mathcal{L}_i$. Similarly, when $(i,j) \in \mathcal{C}$ and $i > j$, then $j \in \mathcal{S}_{ik}$ for all $k \in \mathcal{U}_j$ and $(i,k) \in \mathcal{C}$ for all $k \in \mathcal{U}_j$. Thus, (3.11a) and (3.11b) is a reduced system of equations for the elements of w_{ij} with $(i,j) \in \mathcal{C}$. We can rearrange this system as

$$w_{ij} = g_{ij} - \sum_{k\in\mathcal{L}_i} l_{ki}w_{kj},\ i \le j,\ (i,j)\in\mathcal{C}, \tag{3.12a}$$

$$w_{ij}u_{jj} = g_{ij} - \sum_{k\in\mathcal{U}_j} w_{ik}u_{jk},\ i > j,\ (i,j)\in\mathcal{C}, \tag{3.12b}$$

which can be used to sequentially compute the elements w_{ij} with $(i,j) \in \mathcal{C}$, by computing the columns of $\mathbf{W}_+$ and the rows of $\mathbf{W}_-$, starting with the last column and row. The computational complexity associated with this is about twice the computational complexity associated with calculating the triangular factorization.

An explicit expression for $\mathbf{W}$ can also be derived. The equations (3.9) can be rewritten in the following equivalent form

$$\operatorname{tr}\left[\left(\left(\mathbf{W}\mathbf{U}^T\right)_- + \left(\mathbf{L}^T\mathbf{W}\right)_+\right)^T \mathbf{B}\right] = \operatorname{tr}\left[\mathbf{G}^T\mathbf{B}\right], \qquad \text{for all } \mathbf{B}\in\mathbb{R}^{n\times n}.$$

Using the identity $\operatorname{tr}(\mathbf{XY}) = \operatorname{tr}(\mathbf{YX})$, the right-hand side of this last expression can be reorganized to give

$$\begin{aligned}
\operatorname{tr}\left[\mathbf{G}^T\mathbf{B}\right] &= \operatorname{tr}\left[\mathbf{G}^T\mathbf{L}\mathbf{L}^{-1}\mathbf{B}_- + \mathbf{U}\mathbf{G}^T\mathbf{B}_+\mathbf{U}^{-1}\right] \\
&= \operatorname{tr}\left[\left(\left(\mathbf{L}^T\mathbf{G}\right)_- + \left(\mathbf{G}\mathbf{U}^T\right)_+\right)^T \left(\mathbf{L}^{-1}\mathbf{B}_- + \mathbf{B}_+\mathbf{U}^{-1}\right)\right] \\
&= \operatorname{tr}\left[\mathbf{U}^{-1}\left(\left(\mathbf{L}^T\mathbf{G}\right)_- + \left(\mathbf{G}\mathbf{U}^T\right)_+\right)^T \mathbf{L}^{-1}\left(\mathbf{B}_-\mathbf{U} + \mathbf{L}\mathbf{B}_+\right)\right].
\end{aligned}$$

On the other hand, the left-hand side can be reorganized to give

$$\begin{aligned}
\operatorname{tr}\left[\left(\left(\mathbf{W}\mathbf{U}^T\right)_- + \left(\mathbf{L}^T\mathbf{W}\right)_+\right)^T \mathbf{B}\right] &= \operatorname{tr}\left[\mathbf{U}\mathbf{W}^T\mathbf{B}_- + \mathbf{L}\mathbf{B}_+\right] \\
&= \operatorname{tr}\left[\mathbf{W}^T\left(\mathbf{B}_-\mathbf{U} + \mathbf{L}\mathbf{L}\mathbf{B}_+\right)\right].
\end{aligned}$$

Equating these last two expressions and taking account of the fact that $\mathbf{B}$ is arbitrary, it follows that

$$\mathbf{W}^T = \mathbf{U}^{-1}\left(\left(\mathbf{L}^T\mathbf{G}\right)_- + \left(\mathbf{G}\mathbf{U}^T\right)_+\right)^T \mathbf{L}^{-1}. \tag{3.13}$$

As for direct differentiation, the indirect procedure outlined above is closely related to the backward or adjoint method of algorithmic differentiation (see, for example, Griewank [8] for an overview of algorithmic differentiation or Smith [21] for its application to the Cholesky decomposition). Again, a key difference is that the analysis above is independent of the algorithm used to obtain the triangular factorization. As a consequence, the utilization of the uncoupling of the unknowns in (3.9) (which, again can be implemented in a number of different ways) is far more transparent than would be the case for algorithmic differentiation.

The indirect procedure outlined above has some important advantages when it is required to calculate a number of derivatives. For example, if the components of ∇h need to be calculated, the computation of the components

$$\frac{\partial h(\mathbf{x})}{\partial x_j} = \operatorname{tr}\left(\mathbf{W}^T \frac{\partial \mathbf{A}(\mathbf{x})}{\partial x_j}\right), \qquad j = 1, \ldots, p, \tag{3.14}$$

requires $\mathbf{W}$ to be calculated only once. By contrast, for the direct method, it is necessary to solve an equation of the form (3.4) p times.

When $w(\mathbf{A}) = \log\det\mathbf{A}$, it is clear from (1.4) that $\mathbf{W}^T = \mathbf{A}^{-1}$. This is consistent with (3.13) since it is easy to verify that $(\mathbf{L}^T\mathbf{G})_- + (\mathbf{G}\mathbf{U}^T)_+ = \mathbf{I}$ and so $\mathbf{W}^T = \mathbf{U}^{-1}\mathbf{L}^{-1} = \mathbf{A}^{-1}$. The factors $\mathbf{L}^{-1}$ and $\mathbf{U}^{-1}$ can be written as products of elementary row and column operations, and Niessner and Reichert [18] have used this to propose an algorithm for the efficient calculation of some of the elements of $\mathbf{A}^{-1}$. This algorithm can be easily generalized to the case when $\left(\mathbf{L}^T\mathbf{G}\right)_- + \left(\mathbf{G}\mathbf{U}^T\right)_+$ is a diagonal matrix.

The fact that $\mathbf{W}^T = \mathbf{A}^{-1}$ when $w(\mathbf{A}) = \log\det\mathbf{A}$ provides a link to the work by Takahashi *et al.* [22] and its generalization by Erisman and Tinney [5], on the efficient calculation of some elements of the inverse of a sparse matrix. In fact, (3.9) is essentially the same as equations (1) and (2) in Erisman and Tinney [5]. A more detailed discussion of the various options that can be used to solve the Erisman and Tinney equations was subsequently given by Niessner and Reichert [18].

4. **Second order differentiation**

In this section, we consider the evaluation of $\mathbf{Hc}$, the Hessian of h acting on an arbitrary vector $\mathbf{c}$. The components are

$$\sum_{j=1}^{p} \frac{\partial^2 h}{\partial x_i \partial x_j} c_j = \frac{\partial}{\partial x_i}(\mathbf{c}\cdot\nabla h) = \frac{\partial h'}{\partial x_i}, \qquad i = 1, \ldots, p.$$

4.1. **Direct differentiation.**

Recalling the definition (2.3), inserting $\mathbf{c} = \mathbf{e}_i$ into (3.3) to obtain an equation for $\partial h/\partial x_i$ and then differentiating this equation yields

$$\frac{\partial h'}{\partial x_i} = \operatorname{tr}\left[(\mathbf{G}')^T\left(\frac{\partial \mathbf{L}}{\partial x_i} + \frac{\partial \mathbf{U}}{\partial x_i}\right) + \mathbf{G}^T\left(\frac{\partial \mathbf{L}'}{\partial x_i} + \frac{\partial \mathbf{U}'}{\partial x_i}\right)\right], \tag{4.1}$$

where

$$\mathbf{G}' = \sum_{i>j} \frac{\partial \mathbf{G}}{\partial l_{ij}} l'_{ij} + \sum_{i\leq j} \frac{\partial \mathbf{G}}{\partial u_{ij}} u'_{ij}. \tag{4.2}$$

Despite the apparent complexity of the summation in the first term on the right-hand side of (4.1), it is usually easy to evaluate in practice, when $\mathbf{L}'$, $\mathbf{U}'$, $\frac{\partial \mathbf{L}}{\partial x_i}$ and $\frac{\partial \mathbf{U}}{\partial x_i}$ are known, since it is often the case that only terms of the form $\frac{\partial^2 g}{\partial u_{ii}^2} u'_{ii}$ are nonzero. These first order derivatives can be obtained from equations of the form (3.4) as described in the previous section. The second term on the right-hand side of (4.1) can be calculated as previously once the second derivative term $\frac{\partial \mathbf{L}'}{\partial x_i} + \frac{\partial \mathbf{U}'}{\partial x_i}$ is known. An equation for its determination is obtained by differentiating (3.4); namely,

$$\left(\frac{\partial \mathbf{L}'}{\partial x_i} \mathbf{U} + \mathbf{L} \frac{\partial \mathbf{U}'}{\partial x_i} \right) = \frac{\partial \mathbf{A}'}{\partial x_i} - \frac{\partial \mathbf{L}}{\partial x_i} \mathbf{U}' - \mathbf{L}' \frac{\partial \mathbf{U}}{\partial x_i}. \tag{4.3}$$

The structure of this equation, assuming that the terms on the right-hand side of (4.3) can be easily evaluated when the first derivative terms are known, is the same as in equation (3.4). Consequently, forward substitution can again be applied to sequentially calculate the rows of $\frac{\partial \mathbf{U}'}{\partial x_i}$ and the columns of $\frac{\partial \mathbf{L}'}{\partial x_i}$, starting with the first row and column.

As discussed previously, the computational effort for the calculation of each derivative term is about two times the computational effort required to calculate the factorization. Each of the product terms on the right-hand side of (4.3) requires about the same effort needed to calculate the factorization. The solution of (4.3) requires about two times the effort required to calculate the factorization. Thus, the total effort to calculate one component $\partial h'/\partial x_i$ of the Hessian of h acting on an arbitrary vector $\mathbf{c}$ is about eight times that required to calculate the factorization.

When $\mathbf{c} = \mathbf{e}_j$, $\frac{\partial h'}{\partial x_i} = \frac{\partial^2 h}{\partial x_i \partial x_j}$ is an element of the Hessian $\mathbf{H}$ and these elements must be calculated separately. When $i \neq j$, the computational effort is about eight times that required to calculate the functional, as described above. If, however, $i = j$, then $\frac{\partial h'}{\partial x_i} = \frac{\partial^2 h}{\partial x_i^2}$ and the computational effort is less because the first order derivatives of $\mathbf{L}$ and of $\mathbf{U}$ are the same and the product terms on the right-hand side of (4.3) are the same. It is easy to verify that in this case the computational effort is about five times that required to calculate the functional. If only a single diagonal element of the Hessian is required, for example when $p = 1$, then the direct method outlined above provides an algorithm that is better, in terms of complexity, than the mixed method derived below. However, the mixed method is superior when a number of second order derivatives are required.

4.2. **Mixed methods.** We have seen that the indirect method has the advantage that all of the components of the gradient can be calculated with about the same computational complexity as the evaluation of a single first order derivative using the direct method. It turns out that, by combining the direct and indirect methods, we can also formulate an effective algorithm to calculate the action of the Hessian on an arbitrary vector. For the indirect method, we have

$$h' = \operatorname{tr}\left(\mathbf{W}^T\mathbf{A}'\right) = \sum_{k,l} w_{kl}a'_{kl} \tag{4.4}$$

along with equations (3.9); namely,

$$\left(\mathbf{L}^T\mathbf{W}\right)_+ = \mathbf{G}_+, \qquad \left(\mathbf{W}\mathbf{U}^T\right)_- = \mathbf{G}_-.$$

Recalling the definition (2.3), inserting $\mathbf{c} = \mathbf{e}_i$ into (4.4) to get an equation for $\partial h/\partial x_i$ and then differentiating this equation yields

$$\sum_{j=1}^n \frac{\partial^2 h}{\partial x_i \partial x_j} c_j = \operatorname{tr}\left(\mathbf{W}^T \sum_{j=1}^n \frac{\partial^2 \mathbf{A}}{\partial x_i \partial x_j} c_j\right) + \operatorname{tr}\left((\mathbf{W}')^T \frac{\partial \mathbf{A}}{\partial x_i}\right), \quad i = 1, \ldots, p. \tag{4.5}$$

After computing $\mathbf{W}$ using (3.10a) and (3.10b), the first term on the right-hand side of (4.5) is straightforward to calculate, provided that the second derivatives of $\mathbf{A}$ are easy to calculate. The second term requires $\mathbf{W}'$, and equations for determining it can be obtained by differentiating the equations (3.9). This results in

$$\left(\mathbf{L}^T\mathbf{W}'\right)_+ = \mathbf{G}'_+ - \left((\mathbf{L}')^T\,\mathbf{W}\right)_+, \tag{4.6a}$$

$$\left(\mathbf{W}'\mathbf{U}^T\right)_- = \mathbf{G}'_- - \left(\mathbf{W}\,(\mathbf{U}')^T\right)_-. \tag{4.6b}$$

In addition to $\mathbf{W}$, the right-hand sides of (4.6a) and (4.6b) require $\mathbf{L}'$, $\mathbf{U}'$ and $\mathbf{G}'$. The calculation of $\mathbf{L}'$ and $\mathbf{U}'$ has been described in section 3.1, and because the structure of $g(\mathbf{L}, \mathbf{U})$ is often very simple, the calculation of $\mathbf{G}'$, given $\mathbf{L}'$ and $\mathbf{U}'$ is usually trivial. Each of the product terms on the right-hand sides of (4.6a) and (4.6b) requires about the same effort as an *LU* factorization when $\mathbf{A}$ is a full matrix. Once the right-hand sides are known, equations (4.6a) and (4.6b) can be solved sequentially, starting with the last column and row of $\mathbf{W}'$, with computational complexity of about twice that of the *LU* factorization when $\mathbf{A}$ is full. Consequently, the computational complexity for the calculation of $\mathbf{W}'$ is about eight times that required to perform the *LU* factorization when $\mathbf{A}$ is a full matrix.

When $\mathbf{A}$ is sparse, the computational complexity, relative to *LU* factorization, is substantially greater because $\mathbf{W}$ and $\mathbf{W}'$ are generally full matrices. However, on writing (4.5) in component form

$$\sum_{j=1}^n \frac{\partial^2 h}{\partial x_i \partial x_j} c_j = \sum_{j=1}^n \sum_{l,k=1}^n w_{lk} \frac{\partial^2 a_{lk}}{\partial x_i \partial x_j} c_j + \sum_{l,k=1}^n w'_{lk} \frac{\partial a_{lk}}{\partial x_i}, \; i = 1, \ldots, p,$$

it is clear that we only need to evaluate some of the elements of $\mathbf{W}$ and $\mathbf{W}'$; namely w_{lk} and w'_{lk} with $(l,k) \in \mathcal{C}$. As in section 3.2, it turns out that they can be calculated using a subset of the system of equations given in (4.6a) and (4.6b). Specifically, we have

$$w'_{ij} + \sum_{k \in \mathcal{L}_i} l_{ki} w'_{kj} = g'_{ij} - \sum_{k \in \mathcal{L}_i} l'_{ki} w_{kj}, \; i \le j, \;\; (i,j) \in \mathcal{C}, \tag{4.7a}$$

$$w'_{ij} u_{jj} + \sum_{k \in \mathcal{U}_j} w'_{ik} u_{jk} = g'_{ij} - w_{ij} u'_{jj} - \sum_{k \in \mathcal{U}_j} w_{ik} u'_{jk}, \; i > j, \;\; (i,j) \in \mathcal{C}. \tag{4.7b}$$

As previously, it can be shown that (4.7a) and (4.7b) only involve terms of the form w_{ij} and w'_{ij} with $(i, j) \in \mathcal{C}$. Thus, (4.7a) and (4.7b) is a reduced system of equations for the elements of w'_{ij} with $(i, j) \in \mathcal{C}$. As for the indirect method in section 3.2, the reduced system can be used to sequentially compute the elements w'_{ij} with $(i, j) \in \mathcal{C}$, by computing the columns of $\mathbf{W}'_+$ and the rows of $\mathbf{W}'_-$, starting with the last column and row. The computational complexity associated with implementing the mixed method for a sparse matrix $\mathbf{A}$ is about eight times the computational complexity associated with calculating the LU factorization.

For the evaluation of second derivatives, one could also apply the direct method followed by the indirect method or apply the indirect method twice. Both approaches are equivalent computationally to the mixed method outlined above.

5. Matrix functionals and their derivatives in applications

As mentioned in the Introduction, there are numerous applications in which a matrix functional and its derivatives must be evaluated. Many of these applications have common features such as the specialized structure of the functionals involved.

Reasons for derivatives. In many areas of applied mathematics and statistics, it is natural to formulate an objective functional that depends on some parameters $\mathbf{x}$ through a matrix $\mathbf{A}(\mathbf{x})$. An example is the use of the maximum likelihood method to estimate parameters in a multivariate distribution. The resulting optimization problem is often solved numerically by some form of Newton's method, which requires the evaluation of the gradient and Hessian of the functional (or the action of the Hessian on an arbitrary vector) [17]. If the functional can be evaluated efficiently using a triangular factorization of $\mathbf{A}(\mathbf{x})$, the algorithms outlined in previous sections can greatly reduce the effort for the gradient and Hessian calculations.

Determinant and log determinant functionals. We will see that, in many applications, the functional of interest is the determinant or log determinant of the matrix $\mathbf{A}(\mathbf{x})$.

Determinants arise naturally in applications through the substitution rule for integration. Specifically, if $\mathbf{v} = \mathbf{f}(\mathbf{q})$, where $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$, the differentials transform as $dv_1 \cdots dv_n = |\det(\mathbf{J})|\, dq_1 \cdots dq_n$, where $\mathbf{J}$ is the Jacobian matrix whose elements are $(\mathbf{J})_{ij} = \partial f_i / \partial q_j$. In statistics, an important application of the substitution rule is to determine the probability density function that results when a random variable undergoes a transformation. A simple, and important, example is the multivariate normal distribution

$$f(\mathbf{v}) = \frac{1}{(2\pi)^{n/2} (\det\ \boldsymbol{\Sigma})^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{v} - \boldsymbol{\mu})^T \boldsymbol{\Omega} (\mathbf{v} - \boldsymbol{\mu})\right), \tag{5.1}$$

where $\boldsymbol{\mu}$ is the mean, $\boldsymbol{\Sigma}$ is the positive definite covariance matrix, and $\boldsymbol{\Omega} = \boldsymbol{\Sigma}^{-1}$ is the precision matrix.

There are several identities involving differentiation of the log determinant, which are useful in applications (see e.g. Harville [9]). The elements of the gradient, important in determining the sensitivity of $\mathbf{A}(\mathbf{x})$ with respect to perturbations in the components in the parameters $\mathbf{x}$, can be written as

$$\frac{\partial}{\partial x_k}(\log \det \mathbf{A}) = \operatorname{tr}\left(\mathbf{A}^{-1} \frac{\partial \mathbf{A}}{\partial x_k}\right). \tag{5.2}$$

Setting $x_k = a_{ij}$ in (5.2) yields the identity

$$\frac{\partial \log \det \mathbf{A}}{\partial a_{ij}} = \operatorname{tr}\left(\mathbf{A}^{-1}\mathbf{e}_i\mathbf{e}_j^T\right) = \mathbf{e}_j^T\mathbf{A}^{-1}\mathbf{e}_i, \tag{5.3}$$

which is the starting point for the derivation of many other formulas. The elements of the Hessian, important in a variety of applications including statistics and the application of Newton's method, can be written as

$$\frac{\partial^2}{\partial x_j \partial x_k}(\log \det \mathbf{A}) = \operatorname{tr}\left(\mathbf{A}^{-1}\frac{\partial^2 \mathbf{A}}{\partial x_j \partial x_k} - \mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial x_j}\mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial x_k}\right). \tag{5.4}$$

In a number of applications, including the smoothing spline and A-optimal design applications discussed in the Introduction, there is a scalar parameter x with $\mathbf{A}(x)$ having the linear form $\mathbf{B} + x\mathbf{D}$. In this situation, using induction, the derivatives of the log determinant can be shown to satisfy

$$\frac{d^k \log \det \mathbf{A}(x)}{dx^k} = (-1)^{k-1}(k-1)!\ \operatorname{tr}((\mathbf{A}^{-1}\mathbf{D})^k), \quad k > 0. \tag{5.5}$$

Substituting $\mathbf{D} = \mathbf{C}^T\mathbf{C}$ and k equal to 1 and 2 into (5.5), and using the identity $\operatorname{tr}(\mathbf{XY}) = \operatorname{tr}(\mathbf{YX})$, gives the expressions (1.2a) and (1.2b), respectively. In addition, using the formula (5.5) with $x = \epsilon$ and $\mathbf{A}(\epsilon) = \mathbf{A} + \epsilon\mathbf{I}$ gives the expression (1.3).

In the following subsections, we list several important and novel applications involving the differentiation of matrix functionals, and discuss how the methods in this paper can be utilized to perform efficient calculations.

5.1. **Parameter selection for smoothing splines.** As discussed in the Introduction, in the fitting of splines to noisy data, some criteria for selecting the smoothing parameter require the computation of a functional of the smoothing matrix $S(x)$, where $x > 0$ is the smoothing parameter. In particular, the GCV criterion requires the computation of $\operatorname{tr}(S(x))$ [11, 23], which is the degrees of freedom for the spline, while the RGCV criterion also requires the computation of $\operatorname{tr}(S^2(x))$ [14], which is the residual degrees of freedom [2] (cf. Section 3.5 in [10]). These traces can be evaluated using the equations (1.1), (1.2a) and (1.2b). The methodology of sections 3 and 4 then applies on noting that, for the Cholesky decomposition, $\mathbf{A}(x) = \mathbf{B} + x\mathbf{C}\mathbf{C}^T = \mathbf{L}(x)\mathbf{L}^T(x)$, the differentiation of $\mathbf{A}(x)$ yields $\mathbf{A'} = \mathbf{L'}\mathbf{L}^T + \mathbf{L}(\mathbf{L'})^T = \mathbf{C}\mathbf{C}^T$. Note that the Cholesky decomposition is used here rather than the LU factorization because $\mathbf{A}(x)$ is positive definite.

On occasions, the values of the diagonal elements of $\mathbf{S}(x)$ are required for determining confidence intervals [23]. These can be calculated using

$$\left(\mathbf{C}\mathbf{A}^{-1}\mathbf{C}^T\right)_{ii} = \frac{\partial}{\partial \epsilon_i}\left(\log \det(\mathbf{A} + \mathbf{C}^T \operatorname{diag}(\boldsymbol{\epsilon})\,\mathbf{C})\right)\Big|_{\boldsymbol{\epsilon}=\mathbf{0}}, \ i = 1, 2, \ldots, n+m. \tag{5.6}$$

For the evaluation of the right-hand sides of equations (5.6) and (1.2a), the indirect method is quite efficient as the evaluation of $\mathbf{W}$ only needs to be performed once. However, for the evaluation of the right-hand side of (1.2b), the direct method is more efficient than the mixed method, only requiring about five times the work associated with the triangular factorization compared with eight for the mixed method.

Based on the above approaches, new $O(m^2n)$ algorithms for the evaluation of $\operatorname{tr}(S(x))$ and $\operatorname{tr}(S^2(x))$ are formulated in detail in Lukas *et al.* [15].

5.2. **Covariance selection and graphical methods.** In covariance selection [3] and graphical methods [13, 25] applied to multivariate Gaussian data $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$, as well as other statistical applications, a core activity involves the maximization of the log-likelihood

$$\ell\{\boldsymbol{\Omega}; \mathbf{v}_1, \ldots, \mathbf{v}_n\} = \frac{1}{2} \log\det \boldsymbol{\Omega} + \cdots , \tag{5.7}$$

where the constant, linear terms in $\boldsymbol{\Omega}$ and possible penalty terms have been ignored. In such situations, the precision matrix $\boldsymbol{\Omega}$ is often sparse. The algorithms used to maximize the log-likelihood in (5.7), such as truncated-Newton [17], require elements of the gradient and the action of the Hessian on an arbitrary vector.

In such methods, the vector of unknown parameters is $\mathbf{x} = \mathrm{vec}(\boldsymbol{\Omega})$ and, from (5.3) and (5.4), the gradient and Hessian elements are

$$\frac{\partial}{\partial \boldsymbol{\omega}_{ij}} (\log\det \boldsymbol{\Omega}) = \mathbf{e}_j^T \Sigma \mathbf{e}_i = \sigma_{ji} \tag{5.8}$$

and

$$\frac{\partial^2}{\partial \omega_{lk} \partial \omega_{ij}} (\log\det \boldsymbol{\Omega}) = -\mathbf{e}_j^T \boldsymbol{\Sigma} \frac{\partial \boldsymbol{\Omega}}{\partial \omega_{lk}} \boldsymbol{\Sigma} \mathbf{e}_i = -\sigma_{jl}\sigma_{ki}. \tag{5.9}$$

In addition, the action of the Hessian on any vector $\mathbf{q} = \mathrm{vec}(\mathbf{Q})$ takes the form (5.10)

$$\sum_{i,j} \frac{\partial^2}{\partial \omega_{lk} \partial \omega_{ij}} (\log\det \boldsymbol{\Omega}) q_{ij} = -\sum_{i,j} \sigma_{jl}\sigma_{ki} q_{ij} = -\sum_{i,j} \sigma_{ki} q_{ij} \sigma_{jl} = -\mathbf{e}_k^T \boldsymbol{\Sigma} \boldsymbol{Q} \boldsymbol{\Sigma} \boldsymbol{e}_l.$$

The calculation of this expression reduces to summing only those triple products in (5.10) for which all three entries involved are nonzero. However, even in situations where $\boldsymbol{\Omega}$ is sparse, there is no guarantee that $\boldsymbol{\Sigma Q \Sigma}$ can be evaluated, since the (i, j) component of $\boldsymbol{\Sigma}$ can only be determined from (5.8) for $(i, j) \in \mathcal{C}$. Consequently, it is necessary to turn to the utilization of the mixed method in section 4.2.

5.3. **Restricted maximum likelihood (REML).** In various statistical applications with dependent data, including hierarchical and longitudinal data, it is popular to use the linear mixed model

$$\mathbf{y} = \boldsymbol{X\beta} + \boldsymbol{Zu} + \mathbf{e},$$

where $\boldsymbol{X}$ and $\boldsymbol{Z}$ are incidence matrices, $\boldsymbol{y}$ are observations, $\boldsymbol{\beta}$ are fixed effects parameters, $\boldsymbol{u}$ are random effects parameters and $\boldsymbol{e}$ are random errors. Normally, it is assumed that $E(\boldsymbol{u}) = \mathbf{0}$, $E(\boldsymbol{e}) = 0$, $E\left(\boldsymbol{ue}^T\right) = \mathbf{0}$ and $E\left(\boldsymbol{uu}^T\right) = \boldsymbol{G}(\boldsymbol{\theta})$, $E\left(\boldsymbol{ee}^T\right) = \boldsymbol{R}(\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ denotes the vector of parameters to be estimated. In the REML method [21], θ is estimated by maximizing the log-likelihood

$$\log \mathcal{L} = -\tfrac{1}{2} \left\{\log\det \boldsymbol{R} + \log\det \boldsymbol{G} + \log\det \boldsymbol{C} + \boldsymbol{y}^T \boldsymbol{P} \boldsymbol{y}\right\}, \tag{5.11}$$

where

$$\mathbf{C} = \begin{bmatrix} \mathbf{X}^T\mathbf{R}^{-1}\mathbf{X} & \mathbf{X}^T\mathbf{R}^{-1}\mathbf{Z} \\ \mathbf{Z}^T\mathbf{R}^{-1}\mathbf{X} & \mathbf{Z}^T\mathbf{R}^{-1}\mathbf{Z} + \mathbf{G}^{-1} \end{bmatrix},$$

$$\mathbf{P} = \mathbf{V}^{-1} - \mathbf{V}^{-1}\mathbf{X}\left(\mathbf{X}^T\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{V}^{-1} \text{ and } \mathbf{V} = \mathbf{ZGZ}^T + \mathbf{R}.$$

To perform the maximization efficiently, it is desirable to calculate derivatives of the log-likelihood $\log \mathcal{L}$. According to Smith [21], the terms $\log\det \mathbf{R}$ and $\log\det \mathbf{G}$

are usually easy to evaluate, as are their derivatives. For the evaluation of the other two terms in equation (5.11), it is convenient to consider the mixed model matrix

$$\mathbf{M} = \begin{bmatrix} \mathbf{X}^T\mathbf{R}^{-1}\mathbf{X} & \mathbf{X}^T\mathbf{R}^{-1}\mathbf{Z} & \mathbf{X}^T\mathbf{R}^{-1}\mathbf{y} \\ \mathbf{Z}^T\mathbf{R}^{-1}\mathbf{X} & \mathbf{Z}^T\mathbf{R}^{-1}\mathbf{Z} + \mathbf{G}^{-1} & \mathbf{Z}^T\mathbf{R}^{-1}\mathbf{y} \\ \mathbf{y}^T\mathbf{R}^{-1}\mathbf{X} & \mathbf{y}^T\mathbf{R}^{-1}\mathbf{Z} & \mathbf{y}^T\mathbf{R}^{-1}\mathbf{y} \end{bmatrix},$$

which is a square symmetric, positive definite matrix of dimension n, say. The matrix $\mathbf{C}$, and hence $\mathbf{M}$, are usually very large and very sparse, which is a direct consequence of the fact that both $\mathbf{G}^{-1}$ and $\mathbf{R}^{-1}$ are usually sparse [21]. It is easy to verify [21], that $\mathbf{y}^T\mathbf{P}\mathbf{y} = \det\mathbf{M}/\det\mathbf{C}$. Therefore, if $\mathbf{M} = \mathbf{L}\mathbf{L}^T$ denotes the Cholesky decomposition of $\mathbf{M}$, then

$$\log\det\mathbf{C} + \mathbf{y}^T\mathbf{P}\mathbf{y} = \sum_{k=1}^{n-1}\log\left(l_{kk}^2\right) + l_{nn}^2.$$

Thus, the functional is determined from the triangular decomposition of a sparse matrix. The Cholesky decomposition is used rather than the *LU* factorization because $\mathbf{M}$ is positive definite. Modifications of the analysis in section 3, that take symmetry into account, will yield efficient techniques for calculating derivatives of the last two terms in equation (5.11).

5.4. **Optimal statistical designs.** Suppose we have a vector of n observations

$$\mathbf{y} = \mathbf{C}(\mathbf{x})\boldsymbol{\theta} + \boldsymbol{\delta},$$

where $\mathbf{C}(\mathbf{x}) \in \mathbb{R}^{n\times p}$ is a design matrix that depends on a control vector $\mathbf{x} \in \mathbb{R}^q$, $\boldsymbol{\theta} \in \mathbb{R}^p$ is a vector of unknown parameters that is to be determined and $\boldsymbol{\delta} \in \mathbb{R}^n$ is a vector whose components are independent and normally distributed with variance σ^2. The least squares estimate of the unknown parameters is $\hat{\boldsymbol{\theta}} = \mathbf{C}^+(\mathbf{x})\mathbf{y}$ where $\mathbf{C}^+(\mathbf{x})$ is the Moore-Penrose inverse. For a given design $\mathbf{x}$ and confidence coefficient, the confidence ellipsoid for $\boldsymbol{\theta}$ is given by

$$\left\{\boldsymbol{\theta}\,\middle|\,\left(\boldsymbol{\theta}-\hat{\boldsymbol{\theta}}\right)^T\mathbf{A}(\mathbf{x})\left(\boldsymbol{\theta}-\hat{\boldsymbol{\theta}}\right) \le \text{constant}\right\},$$

where $\mathbf{A}(\mathbf{x}) = \mathbf{C}^T(\mathbf{x})\mathbf{C}(\mathbf{x})$.

The volume of this ellipsoid is proportional to $(\det\mathbf{A}(\mathbf{x}))^{-\frac{1}{2}}$, and it is natural to make this as small as possible. That is, we choose the control vector $\mathbf{x}$ to maximize $\det\mathbf{A}(\mathbf{x})$, which is equivalent to maximizing $\log\det\mathbf{A}(\mathbf{x})$. Such designs are called D-optimal designs (see Silvey [20] for a more detailed discussion). To perform the required optimization, the gradient and Hessian of $\log\det\mathbf{A}(\mathbf{x})$ can be computed efficiently as in the examples on covariance selection and graphical methods in section 5.2.

5.5. **Analysis of short circuit scenarios in power systems.** In the study of power systems, the bus admittance matrix $\mathbf{Y}$ is a complex symmetric matrix whose off-diagonal entries are the negative of the impedances between the buses [22]. As each bus is usually only connected to a few other buses, the admittance matrix is usually quite sparse. For the analysis of short circuits, some of the elements of the bus impedance matrix $\mathbf{Z} = \mathbf{Y}^{-1}$ are required. As explained in [22], not all elements in the impedance matrix are needed if only the distribution of current one bus away from the fault is required. In this case, only certain diagonal elements and elements corresponding to the locally connected branches in the network are required. That is, we require only the elements z_{ji} in the impedance matrix for which the corresponding elements y_{ij} in the admittance matrix are nonzero. Using the basic relationship (5.3), the element z_{ji} can, because of the symmetry, be evaluated as

$$z_{ji} = z_{ij} = \frac{\partial}{\partial y_{ij}} \left(\log \det \mathbf{Y} \right), \qquad \text{for } y_{ij} \neq 0.$$

As explained in section 3.2, the indirect method for evaluating these derivatives yields essentially the same algorithm as that in Takahashi *et al.* [22].

References

[1] R. P. Barry and R. K. Pace. Monte Carlo estimates of the log determinant of large sparse matrices. *J. Linear Algebra Appl.*, 28:41–54, 1999. MR1670972

[2] R. A. Berk. *Statistical Learning from a Regression Perspective.* Springer-Verlag, New York, 2008.

[3] A. Dempster. Covariance selection. *Biometrics*, 28:57–175, 1972.

[4] J. J. Dongarra, I. S. Duff, D. C. Sorensen, and H. A. van der Vorst. *Numerical Linear Algebra for High-Performance Computers.* SIAM Press, Philidelphia, 1998. MR1654239 (2000a:65002)

[5] A. M. Erisman and W. F. Tinney. On computing certain elements of the inverse of a sparse matrix. *Communications ACM*, 18:177–179, 1975. MR0378372 (51:14540)

[6] A. George and J. Li. The evolution of the minimum degree ordering algorithm. *SIAM Review*, 31:1–19, 1989. MR986480 (90g:65033)

[7] G. H. Golub and R. H. Plemmons. Large-scale geodetic least-squares adjustment by dissection and orthogonal decomposition. *Lin. Algebra Appl.*, 34:3–27, 1980. MR591422 (81k:86010)

[8] A. Griewank. *Evaluating Derivatives.* SIAM, Philadelphia, 2000. MR1753583 (2001b:65003)

[9] D. A. Harville. *Matrix Algebra from a Statistician's Perspective.* Springer, New York, 1997. MR1467237 (98k:15001)

[10] T. J. Hastie and R. J. Tibshirani. *Generalized Additive Models.* Chapman and Hall, London, 1990. MR1082147 (92e:62117)

[11] M. F. Hutchinson and F. R. de Hoog. Smoothing noisy data with spline functions. *Numer. Math.*, 47:99–106, 1985. MR797880 (86j:65020)

[12] K. Kubota. Matrix inversion algorithms by means of automatic differentiation. *Appl. Math. Lett.*, 7:19–22, 1994. MR1350388

[13] S. L. Lauritzen. *Graphical Models.* Oxford University Press, New York, 1996. MR1419991 (98g:62001)

[14] M. A. Lukas. Robust generalized cross-validation for choosing the regularization parameter. *Inverse Problems*, 22:1883–1902, 2006. MR2261272 (2007j:62051)

[15] M. A. Lukas, F. R. de Hoog, and R. S. Anderssen. Efficient algorithms for robust generalized cross-validation spline smoothing, *J. Comput. Appl. Math.*, 235:102–107, 2010. MR2671036

[16] H. M. Markowitz. The elimination form of the inverse and its application to linear programming. *Management Sci.*, 3:255–269, 1957. MR0112244 (22:3098)

[17] S. G. Nash. A survey of truncated-Newton methods. *J. Comp. Appl. Math.*, 124:45–59, 2000. MR1803293

[18] H. Niessner and K. Reichert. On computing the inverse of a sparse matrix. *Inter. J. Numer. Methods Eng.*, 19:1513–1526, 1983.

[19] P. Ravikumar, M. J. Wainwright, G. Raskutti, and B. Yu. High-dimensional covariance estimation by minimizing ℓ_1-penalized log-determinant divergence. *IEEE*, 2008.

[20] S. D. Silvey. *Optimal Design.* Chapman and Hall, London, 1980. MR606742 (82d:62123)

[21] S. P. Smith. Differentiation of the Cholesky algorithm. *J. Comp. Graph. Stat.*, 4:134–147, 1995.

[22] K. Takahashi, J. Fagan, and M. S. Chin. Formation of a sparse bus impedance matrix and its application to short circuit study. *Power Industry Computer Applications Conf. Proc. Minneapolis, Minn.*, 8:63–69, June 4-6, 1973.

[23] G. Wahba. Bayesian confidence intervals for the cross-validated smoothing spline. *J.R. Stat. Soc., Ser. B*, 45:133–150, 1983. MR701084 (84k:62054)

[24] F. V. Waugh and P. S. Dwyer. Compact computation of the inverse of a matrix. *Annals Math. Stat.*, 16:259–271, 1945. MR0013919 (7:218d)

[25] J. Whittaker. *Graphical Models in Applied Multivariate Statistics.* Wiley, Chichester, 1990. MR1112133 (93f:62002)

CSIRO Mathematics, Informatics and Statistics, GPO Box 664, Canberra, ACT 2601, Australia

E-mail address: Frank.deHoog@csiro.au

CSIRO Mathematics, Informatics and Statistics, GPO Box 664, Canberra, ACT 2601, Australia

E-mail address: Bob.Anderssen@csiro.au

Mathematics and Statistics, Murdoch University, South Street, Murdoch WA 6150, Australia

E-mail address: M.Lukas@murdoch.edu.au