Evaluating exact VARMA likelihood and its gradient when data are incomplete

KRISTJAN JONASSON
University of Iceland
and
SEBASTIAN E. FERRANDO
Ryerson University

A detailed description of an algorithm for the evaluation and differentiation of the likelihood function for VARMA processes in the general case of missing values is presented. The method is based on combining the Cholesky decomposition method for complete data VARMA evaluation and the Sherman-Morrison-Woodbury formula. Potential saving for pure VAR processes is discussed and formulae for the estimation of missing values and shocks are provided. The paper concludes with description of numerical results obtained with a Matlab implementation of the algorithm, which is in a companion paper. Differentiation with respect to matrices, solution of vector-Yule-Walker equations, VARMA model simulation and the determinant of a low rank update are discussed in appendices.

Categories and Subject Descriptors: G.3 [**Probability and Statistics]**—*Time series analysis*; G.1.3 [**Numerical Analysis]**: Numerical Linear Algebra; G.4 [**Mathematical Software]—***Algorithm design and analysis*; J.2 [**Physical Sciences and Engineering]**; J.4 [**Social and Behavioural Sciences**]—*Economics.*

General Terms: Algorithms.

Additional Key Words and Phrases: Exact likelihood function, missing values, incomplete data, ARMA, VARMA, vector autoregressive moving average model, determinant of low rank update, matrix derivative, matrix differentiation.

1. INTRODUCTION

A key aspect for parameter estimation of autoregressive moving average (ARMA) processes is the efficient evaluation of the likelihood function for the parameters. In practice multivariate or vector-valued processes (VARMA) are important, as well as the more general case of missing values. Evaluation of the *gradient* of the likelihood function is also important for its maximization using traditional numerical optimization.

This paper provides detailed practical formulae for calculating the value and the gradient of a VARMA likelihood function, both for the complete data case, and when there are missing values. A companion algorithm paper [Jonasson 200 xxx] presents Matlab programs that implement these formulae, along with a demonstration on how to carry out actual parameter estimation. Both the ability to deal with missing values and to evaluate gradients are beyond the capabilities of previously published programs. The technique used to treat missing values is also new.

We concentrate on the *exact* likelihood function, not the conditional likelihood (where the initial shocks are assumed to be zero), both because the latter is not easily applicable when values are missing or when the model has moving average terms, and also because the exact likelihood does in many practical applications give significantly better parameter estimates. An alternative to using classical numerical optimization to maximize the likelihood is to use the EM-algorithm, which may in some situations be the method of choice in the presence of missing values. Application of the EM-algorithm to VARMA time series is discussed in a new paper [Metaxoglou and Smith 2007].

Three approaches to calculating exact likelihood of univariate ARMA processes have been described in the literature: (A) one that we shall refer to as the *presample method* described by Siddiqui [1958] for pure MA processes, (B) the *Cholesky decomposition method*, first described by Phadke and Kedem [1978], and (C) a *state space Kalman filter method* described by Harvey and Phillips [1979]. Several authors have described improvements and generalizations of the originally proposed methods, in particular, all three approaches have been generalized to multivariate models and to univariate models with missing values, and the Kalman filter method has been extended to the missing value multivariate case. An overview of the developments is given by Penzer and Shea [1997]. Among the papers discussed there are [Ljung and Box 1979] describing a computationally efficient multivariate implementation of the presample method, [Jones 1980] extending the Kalman filter approach to ARMA with missing values, and [Ansley and Kohn 1983] describing a Kalman filter method to evaluate VARMA likelihood when values are missing. This last method has been publicized in several text books. The Penzer and Shea paper itself deals with extending the Cholesky decomposition method to the univariate missing value case. In addition to the references in [Penzer and Shea 1997], Ljung [1989] discusses estimation of missing values for univariate processes, Mauricio [2002] gives details of a complete data multivariate implementation of the Cholesky decomposition method, and Mélard, Roy and Saidi [2006] describe estimation of structured VARMA models with complete data, allowing unit roots.

Two Fortran programs for VARMA likelihood evaluation in the complete data case have been published: the Kalman filter method is implemented by Shea [1989], and the presample method by Mauricio [1997]. In addition, pure VAR models (with complete data) may be fitted using the Matlab package ARfit, described and published in the pair of papers Neumaier and Schneider [2001] and Schneider and Neumaier [2001].

The Cholesky decomposition method has some advantages. For the complete data case it is considerably simpler and more direct than the other two approaches. Both Penzer and Shea [1997] and Mauricio [2002] compare its efficiency with the Kalman filter method and find that it is faster in the important case when there are more autoregressive terms than moving average terms (cf. Table 1 on p. 925 in Penzer and Shea's paper and Table 1 on p. 484 in Mauricio's paper). As detailed by Penzer and Shea, many authors have also pointed out that for the missing value case the filtering approach may suffer from numerical instabilities, and although remedies have been suggested, they come at some computational cost.

In this paper we take the Cholesky approach. To review its history briefly, the original article of Phadke and Kedem [1978] treats VMA models, extension to ARMA models is in [Ansley 1979], Brockwell and Davis [1987, Ch. 11] describe a VARMA implementation (they and some other authors refer to the method as the *innovation method*) and Penzer and Shea [1997] provide a way of handling missing values in the ARMA case, albeit not the same as our way. To our knowledge, the current paper is the first one to give details of extending the Cholesky decomposition method to the missing value VARMA case, as well as being the first paper to provide derivative formulae. It could be argued that it would have been more useful to give details and an associated publicly available program for the Kalman filter method. According to Ansley and Kohn [1983] missing values do not add to the computational cost of the filtering method, but with the current method many missing values are costly. However few missing values do not cost much, so judging by the results quoted in the previous paragraph our approach wins in that case.

The paper is organized as follows. Section 2 introduces the basic notation and reviews the Cholesky decomposition method for the complete data case. Section 3, the main section of the paper, describes our approach to dealing with the missing value case. Section 4 describes the main ideas and techniques used to compute the derivative of the likelihood function. Section 5 presents some numerical experiments that complement the paper. The appendices present technical material. Appendix A describes our approach to differentiation with respect to matrices, Appendix B describes our solution to the Yule-Walker equations, Appendix C describes how to generate simulated time series, and, finally, Appendix D provides the proof of a result used in the paper.

2. NOTATION AND THE CHOLESKY DECOMPOSITION METHOD

2.1 Model notation

A VARMA model describing a time series of values $\mathbf{x}_t \in \mathbb{R}^r$ for integer t is given by:

$$\mathbf{x}_t - \boldsymbol{\mu} = \sum_{j=1}^{p} A_j(\mathbf{x}_{t-j} - \boldsymbol{\mu}) + \mathbf{y}_t \tag{2.1}$$

where,

$$\mathbf{y}_t = \boldsymbol{\varepsilon}_t + \sum_{j=1}^{q} B_j \boldsymbol{\varepsilon}_{t-j}, \tag{2.2}$$

$\boldsymbol{\mu}$ is the expected value of $\mathbf{x}_t$, the A_j's and the B_j's are $r \times r$ matrices, and the $\boldsymbol{\varepsilon}_t$'s are r-variate $N(\mathbf{0}, \Sigma)$ uncorrelated in time. Let θ denote the $(p+q)r^2 + r(r+3)/2$-dimensional vector of all the parameters (the elements of the A_j's, the B_j's, Σ and $\boldsymbol{\mu}$; Σ being symmetric). If there are no missing values, observations $\mathbf{x}_t$ for $t = 1,\ldots, n$ are given, and $\mathbf{x}$ denotes the nr-vector $(\mathbf{x}_1^{\mathrm{T}},...,\mathbf{x}_n^{\mathrm{T}})^{\mathrm{T}}$ of all these values. When there are missing values the observations are limited to a subvector $\mathbf{x}_{\mathrm{o}} \in \mathbb{R}^N$ of $\mathbf{x}$, and $\mathbf{x}_{\mathrm{m}} \in \mathbb{R}^M$ is a vector of the missing values, say $\mathbf{x}_{\mathrm{m}} = (x_{m_1},...,x_{m_M})$. If the time series is stationary then the complete data log-likelihood function is given by

$$l(\theta) = -\tfrac{1}{2}\left(nr\log 2\pi + \log\det S + (\mathbf{x} - \bar{\boldsymbol{\mu}})^{\mathrm{T}} S^{-1}(\mathbf{x} - \bar{\boldsymbol{\mu}})\right) \tag{2.3}$$

where $S = \operatorname{cov}_\theta(\mathbf{x})$ and $\bar{\boldsymbol{\mu}} = \mathrm{E}_\theta(\mathbf{x}) = (\boldsymbol{\mu}^{\mathrm{T}},\ldots,\boldsymbol{\mu}^{\mathrm{T}})^{\mathrm{T}}$. The log-likelihood function for the observed data is given by

$$l_{\mathrm{o}}(\theta) = -\tfrac{1}{2}\left(N\log 2\pi + \log\det S_{\mathrm{o}} + (\mathbf{x}_{\mathrm{o}} - \bar{\boldsymbol{\mu}}_{\mathrm{o}})^{\mathrm{T}} S_{\mathrm{o}}^{-1}(\mathbf{x}_{\mathrm{o}} - \bar{\boldsymbol{\mu}}_{\mathrm{o}})\right) \tag{2.4}$$

where $S_{\mathrm{o}} = \operatorname{cov}_\theta(\mathbf{x}_{\mathrm{o}})$ is obtained from S by removing rows $m_1,\ldots, m_M$ and columns $m_1,\ldots, m_M$, and $\bar{\boldsymbol{\mu}}_{\mathrm{o}}$ = $\mathrm{E}_\theta(\mathbf{x}_{\mathrm{o}})$ is obtained from $\bar{\boldsymbol{\mu}}$ by removing components $m_1,\ldots, m_M$ (see for example [Ljung 1989]).

We have included the mean of the series among the parameters, instead of assuming a zero-mean process as is customary in the literature. This is not important when there are no missing values: one can simply subtract the mean of the series. When there are missing values, this might however cause a bias. Say a weather station was out of function during a cold spell. Then the mean of all observed temperature values would probably overestimate the true mean, but if other nearby stations were measuring during the cold spell then maximizing the likelihood of a VARMA model with the mean as a free parameter would avoid this bias.

2.2 Likelihood evaluation for complete data

We now turn attention to the evaluation of (2.3) and proceed in a similar vein as Mauricio [2002] and Brockwell and Davis [1987] (and as briefly suggested in [Penzer and Shea 1997]). From (2.1),

$$\mathbf{y}_t = \mathbf{x}_t - \boldsymbol{\mu} - \textstyle\sum_{j=1}^{p} A_j(\mathbf{x}_{t-j} - \boldsymbol{\mu})$$

for $t > p$. Let $\mathbf{w}_t$ = $\mathbf{x}_t - \boldsymbol{\mu}$ for $t \le p$ and $\mathbf{w}_t$ = $\mathbf{y}_t$ for $t > p$ and let $\mathbf{w}$ = $(\mathbf{w}_1^{\mathrm{T}},...,\mathbf{w}_n^{\mathrm{T}})^{\mathrm{T}}$. Then w = $\Lambda(\mathbf{x} - \bar{\boldsymbol{\mu}})$ where Λ is the $nr \times nr$ lower triangular block-band matrix given by

$$\Lambda = \begin{bmatrix} I & & & & & & \\ & \ddots & & & & & \\ & & I & & & & \\ -A_p & \cdots & -A_1 & I & & & \\ & \ddots & & \ddots & \ddots & & \\ & & -A_p & \cdots & -A_1 & I \end{bmatrix}. \tag{2.5}$$

Now let $C_j = \operatorname{cov}(\mathbf{x}_t, \boldsymbol{\varepsilon}_{t-j})$, $G_j = \operatorname{cov}(\mathbf{y}_t, \mathbf{x}_{t-j})$, $W_j = \operatorname{cov}(\mathbf{y}_t, \mathbf{y}_{t-j})$ and $S_j = \operatorname{cov}(\mathbf{x}_t, \mathbf{x}_{t-j})$, (all these are $r \times r$ matrices). Note that with this notation,

$$S = \begin{bmatrix} S_0 & S_1^{\mathrm{T}} & \cdots & S_{n-1}^{\mathrm{T}} \\ S_1 & S_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & S_1^{\mathrm{T}} \\ S_{n-1} & \cdots & S_1 & S_0 \end{bmatrix}. \tag{2.6}$$

Furthermore, let A_i and B_j be zero for i and j outside the ranges implied by (2.1). By multiplying through (2.1) from the right with $\boldsymbol{\varepsilon}_{t-j}^{\mathrm{T}}$ for $j = 0,\ldots, q$ and taking expectations the following recurrence formulae for C_0, C_1, C_2,… are obtained:

$$C_j = A_1 C_{j-1} + \ldots + A_j C_0 + B_q \Sigma, \text{ for } j = 0, 1,\ldots \tag{2.7}$$

(so $C_0 = \Sigma$). With $B_0 = I$, we have by (2.1) and (2.2):

$$G_j = B_j C_0^{\mathrm{T}} + \ldots + B_q C_{q-j}^{\mathrm{T}}, \text{ for } j = 0,\ldots, q, \tag{2.8}$$

$$W_j = B_j \Sigma B_0^{\mathrm{T}} + \ldots + B_q \Sigma B_{q-j}^{\mathrm{T}}, \text{ for } j = 0,\ldots, q. \tag{2.9}$$

For $j < 0$ or $j > q$, C_j, G_j and W_j are zero. By multiplying (2.1) from the right with $(\mathbf{x}_{t-j} - \boldsymbol{\mu})^{\mathrm{T}}$ for $j = 0,\ldots, p$ and taking expectations one gets the following linear system (the *vector-Yule-Walker* equations) for the $r(r+1)/2 + pr^2$ elements of $S_0,\ldots, S_p$ (note that S_0 is symmetric):

$$\begin{aligned} S_0 - A_1 S_1^{\mathrm{T}} - \ldots - A_p S_p^{\mathrm{T}} &= G_0 \\ S_1 - A_1 S_0 - A_2 S_1^{\mathrm{T}} - \ldots - A_p S_{p-1}^{\mathrm{T}} &= G_1 \\ S_2 - A_1 S_1 - A_2 S_0 - A_3 S_1^{\mathrm{T}} - \ldots - A_p S_{p-2}^{\mathrm{T}} &= G_2 \\ &\vdots \\ S_p - A_1 S_{p-1} - A_2 S_{p-2} - \ldots - A_p S_0 &= G_p \end{aligned} \tag{2.10}$$

The solution of (2.10) is dealt with in Appendix B. If $q \le p$, the covariance matrix of $\mathbf{w}$ will be given by the $nr \times nr$ matrix:

$$\Omega = \left[\begin{array}{cccc|cccccc} S_0 & S_1^{\mathrm{T}} & \cdots & S_{p-1}^{\mathrm{T}} & & & & & & \\ S_1 & S_0 & \ddots & \vdots & G_q^{\mathrm{T}} & & & & & \\ \vdots & \ddots & \ddots & S_1^{\mathrm{T}} & \vdots & \ddots & & & & \\ S_{p-1} & \cdots & S_1 & S_0 & G_1^{\mathrm{T}} & \cdots & G_q^{\mathrm{T}} & & & \\ \hline & G_q & \cdots & G_1 & W_0 & W_1^{\mathrm{T}} & \cdots & W_q^{\mathrm{T}} & & \\ & & \ddots & \vdots & W_1 & W_0 & W_1^{\mathrm{T}} & & \ddots & \\ & & & G_q & \vdots & W_1 & W_0 & \ddots & & W_q^{\mathrm{T}} \\ & & & & W_q & & \ddots & \ddots & W_1^{\mathrm{T}} & \vdots \\ & & & & & \ddots & & W_1 & W_0 & W_1^{\mathrm{T}} \\ & & & & & & W_q & \cdots & W_1 & W_0 \end{array}\right] \tag{2.11}$$

If $q > p$ the depiction is slightly different, the $pr \times (n-p)r$ upper right partition of Ω will be

$$\begin{bmatrix} G_p^{\mathrm{T}} & \cdots & G_q^{\mathrm{T}} & & & \\ \vdots & & & \ddots & & \\ G_1^{\mathrm{T}} & G_2^{\mathrm{T}} & \cdots & \cdots & G_q^{\mathrm{T}} & \end{bmatrix},$$

the lower left partition will be the transpose of this, but the upper left and lower right partitions are unaltered. Since Λ has unit diagonal, one finds that

$$l(\theta) = -\tfrac{1}{2}\left(nr\log 2\pi + \log\det\Omega + \mathbf{w}^{\mathrm{T}}\Omega^{-1}\mathbf{w}\right) \tag{2.12}$$

To evaluate (2.12) it is most economical to calculate the Cholesky-factorization $\Omega = LL^{\mathrm{T}}$ exploiting the block-band structure and subsequently determine $\mathbf{z} = L^{-1}\mathbf{w}$ using forward substitution. Then the log-likelihood function will be given by

$$l(\theta) = -\tfrac{1}{2}(nr\log 2\pi + 2\textstyle\sum_i \log l_{ii} + \mathbf{z}^{\mathrm{T}}\mathbf{z}). \tag{2.13}$$

We remark that the exposition in [Brockwell and Davis 1987] is significantly different from ours. They talk of the *innovation* algorithm but it turns out that the actual calculations are identical to the Cholesky decomposition described here.

2.3 Operation count for complete data

Let $h = \max(p,q)$ and assume that $q > 0$ (see Section 3.4 for the $q = 0$ case). Given $\mathbf{x}$ it takes $r^2 p(n-p)$ multiplications to calculate $\mathbf{w}$. Determining the C_j's for $j \le q$, G_j's and W_i's with (2.7), (2.8) and (2.9) costs about $r^3(\min(p,q)^2/2 + q^2)$ multiplications and solving the system (2.10) takes roughly $r^6 p^3/3$ multiplications. The cost of the Cholesky-factorization of Ω will be about $(rh)^3/6$ multiplications for the upper left partition and $r^3(n-h)(q^2/2 + 7/6)$ for the lower partition. Finally, the multiplication count for the forward substitution for $\mathbf{z}$ is about $r^2(h^2/2 + (p/2+q)(n-h))$.

To take an example of the savings obtained by using (2.13) rather than (2.3) let $p = q = 3$, $r = 8$ and $n = 1000$. Then Cholesky-factorization of S will cost $8000^3/6 \approx 8.5\cdot 10^{10}$ multiplications (and take about 7 min. on a 1600 MHz Pentium M processor) but calculation with (2.13), including all the steps leading to it, will take $4.0\cdot 10^6$ multiplications (and take 0.02 s).

3. MISSING VALUE CASE

3.1 Likelihood evaluation via the Sherman-Morrison-Woodbury formula

We now consider the economical evaluation of (2.4) in the presence of some missing observations. As is customary, we assume that observations "are missing at random", i.e. that whether an observation is missing does not depend on its numerical value.

Consider first the term $(\mathbf{x}_{\mathrm{o}} - \bar{\boldsymbol{\mu}}_{\mathrm{o}})^{\mathrm{T}} S_{\mathrm{o}}^{-1}(\mathbf{x}_{\mathrm{o}} - \bar{\boldsymbol{\mu}}_{\mathrm{o}})$. Let $\bar{\Omega}$, $\bar{\Lambda}$ and $\bar{S}$ be obtained from Ω, Λ and S by placing rows and columns $m_1,\ldots,m_M$ after the other rows and columns and partition them as follows (with Ω_{o}, Λ_{o} and S_{o} being $N \times N$, and Ω_{m}, Λ_{m} and S_{m} being $M \times M$):

$$\bar{\Omega} = \begin{bmatrix} \Omega_{\mathrm{o}} & \Omega_{\mathrm{om}} \\ \Omega_{\mathrm{mo}} & \Omega_{\mathrm{m}} \end{bmatrix},\ \bar{\Lambda} = \begin{bmatrix} \Lambda_{\mathrm{o}} & \Lambda_{\mathrm{om}} \\ \Lambda_{\mathrm{mo}} & \Lambda_{\mathrm{m}} \end{bmatrix},\ \text{and } \bar{S} = \begin{bmatrix} S_{\mathrm{o}} & S_{\mathrm{om}} \\ S_{\mathrm{mo}} & S_{\mathrm{m}} \end{bmatrix}.$$

By the definition of $\mathbf{w}$, $\Omega = \Lambda S \Lambda^{\mathrm{T}}$ and therefore

$$\Omega_{\mathrm{o}} = \Lambda_{\mathrm{o}} S_{\mathrm{o}} \Lambda_{\mathrm{o}}^{\mathrm{T}} + \Lambda_{\mathrm{o}} S_{\mathrm{om}} \Lambda_{\mathrm{om}}^{\mathrm{T}} + \Lambda_{\mathrm{om}} S_{\mathrm{mo}} \Lambda_{\mathrm{o}}^{\mathrm{T}} + \Lambda_{\mathrm{om}} S_{\mathrm{m}} \Lambda_{\mathrm{om}}^{\mathrm{T}}. \tag{3.1}$$

Λ_{o} is obtained from Λ by removing rows and corresponding columns, and it is therefore an invertible lower band matrix with unit diagonal and bandwidth at most rp, and Ω_{o} is obtained from Ω by removing rows and corresponding columns, so it is also a band matrix and its triangular factorization will be economical. It is thus attractive to operate with these matrices rather than the full matrix S_{o}. Defining

$$\tilde{\Omega}_{\mathrm{o}} = \Lambda_{\mathrm{o}} S_{\mathrm{o}} \Lambda_{\mathrm{o}}^{\mathrm{T}} \tag{3.2}$$

and $\tilde{\mathbf{w}}_{\mathrm{o}} = \Lambda_{\mathrm{o}}(\mathbf{x}_{\mathrm{o}} - \bar{\boldsymbol{\mu}})$ we have $(\mathbf{x}_{\mathrm{o}} - \bar{\boldsymbol{\mu}}_{\mathrm{o}})^{\mathrm{T}} S_{\mathrm{o}}^{-1}(\mathbf{x}_{\mathrm{o}} - \bar{\boldsymbol{\mu}}_{\mathrm{o}}) = \tilde{\mathbf{w}}_{\mathrm{o}}^{\mathrm{T}} \tilde{\Omega}_{\mathrm{o}}^{-1} \tilde{\mathbf{w}}_{\mathrm{o}}$. Also, from (3.1) and (3.2)

$$\tilde{\Omega}_{\mathrm{o}} = \Omega_{\mathrm{o}} - \Lambda_{\mathrm{o}} S_{\mathrm{om}} \Lambda_{\mathrm{om}}^{\mathrm{T}} - \Lambda_{\mathrm{om}} S_{\mathrm{om}}^{\mathrm{T}} \Lambda_{\mathrm{o}}^{\mathrm{T}} - \Lambda_{\mathrm{om}} S_{\mathrm{m}} \Lambda_{\mathrm{om}}^{\mathrm{T}}, \tag{3.3}$$

(keep in mind that $\bar{S}$ is symmetric). The matrices S_{om}, Λ_{om} and $S_{\mathrm{o}}\Lambda_{\mathrm{om}}$ are $N \times M$, so if the number of missing values, M, is (considerably) smaller than the number of observations, N, then (3.3) represents a low rank modification of Ω_{o}. This invites the use of the Sherman-Morrison-Woodbury (SMW) formula [Sherman and Morrison 1950; Woodbury 1950; c.f. Golub and Van Loan 1983]. To retain symmetry of the matrices that need to be factorized, (3.3) may be rewritten as:

$$\tilde{\Omega}_{\mathrm{o}} = \Omega_{\mathrm{o}} + US_{\mathrm{m}}^{-1}U^{\mathrm{T}} - VS_{\mathrm{m}}^{-1}V^{\mathrm{T}} \tag{3.4}$$

where $U = \Lambda_{\mathrm{o}}S_{\mathrm{om}}$ and $V = \Lambda_{\mathrm{o}}S_{\mathrm{om}} + \Lambda_{\mathrm{om}}S_{\mathrm{m}}$. It turns out that U is generally a full matrix but V is sparse, and it will transpire that it is possible to avoid forming U.

To obtain V economically, select the observed rows and missing columns from ΛS. From (2.5), (2.6) and proceeding as when deriving (2.10) the following block representation of ΛS for the case $q > p$ is obtained:

$$\Lambda S = \left[\begin{array}{ccccccc}
S_0 & S_1^{\mathrm{T}} & \cdots & & & & S_{n-1}^{\mathrm{T}} \\
\vdots & \ddots & & & & & \vdots \\
S_{p-1} & \cdots & S_0 & S_1^{\mathrm{T}} & \cdots & & S_{n-p}^{\mathrm{T}} \\
\hline
G_p & \cdots & G_1 & G_0 & G_{-1} & \cdots & G_{-n+p+1} \\
\vdots & & & & \ddots & & \vdots \\
G_q & & & & & & \vdots \\
 & \ddots & & & & \ddots & G_{-1} \\
 & & G_q & \cdots & & & G_0
\end{array}\right].$$

For $q \le p$ the upper partition is the same but the lower partition is:

$$\left[\begin{array}{ccccccc}
 & G_q & \cdots & G_0 & \cdots & & G_{-n+p+1} \\
 & & \ddots & & \ddots & & \vdots \\
 & & & \ddots & & \ddots & \vdots \\
 & & & & G_q & \cdots & G_0
\end{array}\right].$$

For $S_{p+1}, \ldots, S_{n-1}$, multiply (2.1) from the right with $(\mathbf{x}_{t-j} - \boldsymbol{\mu})^{\mathrm{T}}$ for $j = p+1, \ldots, n-1$ and take expectations (as when deriving (2.10)), giving

$$S_j = A_1 S_{j-1} + A_2 S_{j-2} + \ldots + A_p S_{j-p} + G_j \tag{3.5}$$

with $G_p = 0$ for $p > q$. The G_j for negative j may be obtained using:

$$G_{-j} = C_j^{\mathrm{T}} + B_1 C_{j+1}^{\mathrm{T}} + \ldots + B_q C_{j+q}^{\mathrm{T}}$$

where the C_i's are given by the recurrence (2.7).

From (2.10) and (3.5) it follows that blocks (i, j) with $i > j + q$ of ΛS are zero, giving almost 50% sparsity. If the missing values are concentrated near the beginning of the observation period then V will be sparser still. This applies, for example, when the series represent measurements that did not all start at the same time, as will often be the case in practice. To take an example, if $q = 1$, $r = 2$, $n = 6$ and $\mathbf{m} = (2, 3, 4, 5, 9)$ then the sparsity pattern of V will be:

$$\left[\begin{array}{ccccc}
\times & \times & \times & \times & \times \\
 & \times & \times & \times & \times \\
 & & & \times & \times \\
 & & & & \times \\
 & & & & \times \\
 & & & & \times \\
 & & & & \times
\end{array}\right]$$

The SMW formula applied to (3.4) gives

$$\tilde{\Omega}_o^{-1} = \hat{\Omega}_o^{-1} + \hat{\Omega}_o^{-1} V Q^{-1} V^T \hat{\Omega}_o^{-1}$$

where

$$\hat{\Omega}_o = \Omega_o + U S_m^{-1} U^T \tag{3.6}$$

and Q is the $M \times M$ matrix $S_m - V^T \hat{\Omega}_o^{-1} V$. Moreover, if R is the $M \times M$ matrix $S_m + U^T \Omega_o^{-1} U$ then (again by the SMW formula):

$$\hat{\Omega}_o^{-1} = \Omega_o^{-1} - \Omega_o^{-1} U R^{-1} U^T \Omega_o^{-1}.$$

If L_R is the Cholesky factor of R and $K = L_R^{-1} U^T \Omega_o^{-1} V$ it follows that $Q = S_m - V^T \Omega_o^{-1} V + K^T K$. The first method that springs to mind to evaluate $V^T \Omega_o^{-1} V$ efficiently is to Cholesky factorize $\Omega_o = LL^T$, use forward substitution to obtain $\hat{V} = L^{-1} V$ and form $\hat{V}^T \hat{V}$. However, with this procedure $\hat{V}$ will be full and the computation of $\hat{V}^T \hat{V}$ will cost $NM(M+1)/2$ multiplications. In contrast, if an $L^T L$-factorization of Ω_o is employed instead of an LL^T-factorization the sparsity of V will be carried over to $\hat{V}$ with large potential savings. This is a crucial observation because, with many missing values, multiplication with $\hat{V}$ constitutes the bulk of the computation needed for the likelihood evaluation.

Thus the proposed method is: $L^T L$-factorize $\Omega_o = L_o^T L_o$, and back-substitute to get $\hat{V} = L_o^{-T} V$ and $\hat{\Lambda}_{om} = L_o^{-T} \Lambda_{om}$, making use of known sparsity for all calculations (the sparsity structure of $\hat{\Lambda}_{om}$ will be similar to that of $\hat{V}$). With $R_V = \hat{V}^T \hat{V}$, $R_\Lambda = \hat{\Lambda}_{om}^T \hat{\Lambda}_{om}$ and $P = \hat{\Lambda}_{om}^T \hat{V}$ (again exploiting sparsity) we find that $R = S_m + R_V + S_m R_\Lambda S_m - S_m P - P^T S_m$ (all matrices in this identity are full $M \times M$), $K = L_R^{-1}(R_V - P)$ and $Q = S_m - R_V + K^T K$. Let further L_Q be the Cholesky factor of Q, $\hat{\mathbf{w}}_o = L_o^{-1} \tilde{\mathbf{w}}_o$, $\mathbf{u} = L_R^{-1}(\hat{V}^T \hat{\mathbf{w}}_o - S_m \hat{\Lambda}_{om}^T \hat{\mathbf{w}}_o)$ and $\mathbf{v} = L_Q^{-1}(\hat{V}^T \hat{\mathbf{w}}_o - K^T \mathbf{u})$. A little calculation then gives:

$$(\mathbf{x}_o - \bar{\boldsymbol{\mu}}_o)^T S_o^{-1} (\mathbf{x}_o - \bar{\boldsymbol{\mu}}_o) = \hat{\mathbf{w}}_o^T \hat{\mathbf{w}}_o - \mathbf{u}^T \mathbf{u} + \mathbf{v}^T \mathbf{v}.$$

Now turn attention to the other nontrivial term in (2.4), $\log\det S_o$. From $\det(I + AB) = \det(I + BA)$ (see Appendix D) we get $\det(X \pm AY^{-1}A^T) = \det X \det Y^{-1} \det(X \pm A^T Y^{-1} A)$. From (3.3), (3.6) and the definition of L_Q, $\det \tilde{\Omega}_o = \det(\hat{\Omega} - V S_m^{-1} V^T) = \det \hat{\Omega}_o \det S_m^{-1} \det(S_m - V^T \hat{\Omega}_o^{-1} V) = \det \hat{\Omega}_o \det S_m^{-1} \det(L_Q)^2$. Similarly, $\det \hat{\Omega}_o = \det(\Omega_o + U S_m^{-1} U^T) = \det \Omega_o \det S_m^{-1} \det(S_m + U^T \Omega_o^{-1} U) = \det \Omega_o \det S_m^{-1} \det(L_R)^2$. Since $\det \Lambda_o = 1$ it now follows from (3.2) and the definition of L_o that

$$\log\det S_o = 2(\log\det L_o + \log\det L_R + \log\det L_Q - \log\det S_m).$$

3.2 Estimating missing values and shocks

An obvious estimate of the vector of missing values is its expected value, $\mathbf{x}_m^E = \mathrm{E}(\mathbf{x}_m \mid \mathbf{x}_o, \theta)$, where θ is the maximum likelihood estimate of the parameters (this is also the maximum likelihood estimate of $\mathbf{x}_m$). Since $S_{mo} = \mathrm{cov}(\mathbf{x}_m, \mathbf{x}_o)$ and $S_o = \mathrm{var}(\mathbf{x}_o)$,

$$\mathbf{x}_m^E = S_{mo} S_o^{-1} (\mathbf{x}_o - \bar{\boldsymbol{\mu}}_o) + \bar{\boldsymbol{\mu}}_m.$$

(where $\bar{\boldsymbol{\mu}}_m$ consists of missing components of $\bar{\boldsymbol{\mu}}$). Similarly, the maximum likelihood estimate of the shocks $\boldsymbol{\varepsilon}_t$ is given by $\boldsymbol{\varepsilon}_t^E = \mathrm{E}(\boldsymbol{\varepsilon}_t \mid \mathbf{x}_o, \theta)$. For $0 \le j \le q$, $\mathrm{cov}(\boldsymbol{\varepsilon}_t, \mathbf{x}_{t+j}) = C_j^T$ and $\boldsymbol{\varepsilon}_t$ is independent of $\mathbf{x}_{t+j}$ for other j. It follows that $\boldsymbol{\varepsilon}^E = \tilde{C} S_o^{-1} (\mathbf{x}_o - \bar{\boldsymbol{\mu}}_o)$ where $\boldsymbol{\varepsilon}^E$ is the column vector with $\boldsymbol{\varepsilon}_1^E, \ldots, \boldsymbol{\varepsilon}_n^E$ and $\tilde{C}$ is obtained by removing missing columns from the $nr \times nr$ matrix:

$$\begin{bmatrix} C_0 & C_1^T & \cdots & C_q^T & & \\ & C_0 & & & \ddots & \\ & & \ddots & & & C_q^T \\ & & & \ddots & & \vdots \\ & & & & C_0 & C_1^T \\ & & & & & C_0 \end{bmatrix}.$$

With some calculation one may verify that given the matrices and vectors defined in the previous section, the estimates of $\mathbf{x}_\mathrm{m}$ and $\boldsymbol{\varepsilon}$ may be calculated economically using:

$$\mathbf{x}_\mathrm{m}^E = S_\mathrm{m}\mathbf{v}_2 + \bar{\boldsymbol{\mu}}_\mathrm{m},$$

and

$$\boldsymbol{\varepsilon}^E = \tilde{C}\Lambda_\mathrm{o}^\mathrm{T} L_\mathrm{o}^{-\mathrm{T}}(\hat{\mathbf{w}}_\mathrm{o} + \hat{V}(\mathbf{v}_1 - \mathbf{v}_2) - \hat{\Lambda}_\mathrm{om} S_\mathrm{m}\mathbf{v}_2)$$

where $\mathbf{v}_1 = L_Q^{-\mathrm{T}}\mathbf{v}$ and $\mathbf{v}_2 = L_R^{-\mathrm{T}}(\mathbf{u} + K\mathbf{v}_1)$.

3.3 Simplification for pure autoregressive models

If q is zero and there are no moving average terms considerable simplification results, and it is worthwhile to review this case. Since $\mathbf{y}_t = \boldsymbol{\varepsilon}_t$ for all t the G_j and W_j matrices will all be zero apart from G_0 and W_0, which are both equal to Σ. The upper left S-partition of Ω in (2.11) will be unchanged, the G-partition will be zero and the lower-right W-partition will be a block diagonal matrix where each block is equal to Σ. For the missing value case, Ω_o needs to be Cholesky factorized. It is obtained by removing rows and corresponding columns from Ω, so that its upper left partition is the same as in the general ARMA case, but the lower right partition is a block diagonal matrix:

$$\begin{bmatrix} \Sigma_{\mathrm{o}1} & & & \\ & \Sigma_{\mathrm{o}2} & & \\ & & \ddots & \\ & & & \Sigma_{\mathrm{o},n-p} \end{bmatrix}$$

where $\Sigma_{\mathrm{o}i}$ contains rows and columns of Σ corresponding to the observed indices at time $p + i$. To obtain L_o it is therefore sufficient to Cholesky factorize $\Sigma_{\mathrm{o}i}$ for each missing pattern that occurs, which in all realistic cases will be much cheaper than Cholesky factorizing the entire Ω_o-matrix.

3.4 Operation count for missing value likelihood

Finding the C_j's, G_j's and W_j's and S_j's will be identical to the complete data case. The Cholesky factorization of Ω_o costs at most $r^2N(q^2/2 + 7/6)$ multiplications (unless the upper left partition is unusually big). Forming ΛS costs about $r^3(2p + q)(n - p)$ multiplications. The cost of forming $\hat{\Lambda}_\mathrm{om}$ and $\hat{V}$ using back substitution depends on the missing value pattern. In the worst case, when all the missing values are at the end of the observation period the cost is approximately $rqNM$ multiplications for each, since the bandwidth of both is $\leq rq$, but as explained in Section 3.1 the missing values will often be concentrated near the beginning and the cost will be much smaller. The cost of R_V, R_Λ and P also depends on the missing value pattern. In the worst case the symmetric R_V and R_Λ cost $NM^2/2$ multiplications each and P costs NM^2, but the typical cost is again much smaller (for example, with the "miss-25" pattern of Table I the cost is 5 times smaller). Next follows a series of order M^3 operations: $S_\mathrm{m}P$ costs M^3, R costs $3M^3/2$, K and Q cost $M^3/2$ multiplications each. Finally the Cholesky factorizations for each of L_R, L_Q and $\det S_\mathrm{m}$ cost $M^3/6$ multiplications. The multiplication count of other calculations is negligible by comparison unless M is very small. When n and M are large compared to p, q and r the governing tasks will cost $2fNM^2 + 4M^2$ multiplications where f is the savings factor of having the missing values early on.

In the pure autoregressive case the C_j's, G_j's and W_j's come for free, but solving the vector-Yule-Walker equations costs the same as before. The cost of Cholesky factorizing Ω will usually be negligible, and much cheaper than when $q > 0$. When nothing is missing, it is the number rpN of multiplications to find $\mathbf{w}$ and the number $rpN/2$ of multiplications of the forward substitution for $\mathbf{z}$ that govern the computational cost. On the negative side, there will be no savings in the governing tasks when M and n are large.

4. DERIVATIVE OF THE LIKELIHOOD FUNCTION

Several different matrix operations that need to be differentiated may be identified. Matrix products are used in the calculation of **w** and the covariance matrices C_i, G_i and W_i, Cholesky factorization gives **Ω**, linear equations are solved to obtain the S_i-matrices and **z**, and lastly one must differentiate $\log\det L$. In the missing value case, several more matrix products, Cholesky factorizations, linear equation solutions and determinants occur.

Nel [1980] reviews and develops matrix differentiation methods of scalar and matrix-valued functions with respect to scalars and matrices. He discusses three basic methods, and concludes that a method that he calls the *element breakdown* method is best for general purposes, and this is the approach we take. For the change of variables described in Section 4.3 we also make use of his *vector rearrangement* method.

Since there is no commonly used notation for differentiation with respect to matrices, we provide the needed notation and formulae in Appendix A for clarity and ease of reference.

4.1 Derivatives of the r × r covariance matrices

The matrices C_i, G_i and W_i are all simple matrix-polynomials in the parameter matrices (the A_i's, B_i's and Σ), and it is not difficult to verify that they can all be obtained by applying a sequence of operations of the following types:

$$\begin{aligned} F &\leftarrow F + XY \\ F &\leftarrow F + XY^{\mathrm{T}} \\ F &\leftarrow F + XG \\ F &\leftarrow F + XG^{\mathrm{T}} \end{aligned} \tag{4.1}$$

where F is the polynomial, X and Y are independent variables (parameter matrices), and G is also a polynomial obtained through such steps. Initialization can be either $F \leftarrow O$ (the $r \times r$ zero matrix) or $F \leftarrow X$ (one of the parameter matrices). The operations (4.1) can all be differentiated using (A.3) and (A.4) as detailed in the following table, where X, Y and Z are different parameter matrices:

Change to F:	Corresponding change to:		
	$[dF/dZ]_{lc}$	$[dF/dX]_{lc}$	$[dF/dY]_{lc}$
$+XY$	0	$+\mathbf{e}_l\mathbf{e}_c^{\mathrm{T}}Y$	$+X\mathbf{e}_l\mathbf{e}_c^{\mathrm{T}}$
$+XY^{\mathrm{T}}$	0	$+\mathbf{e}_l\mathbf{e}_c^{\mathrm{T}}Y^{\mathrm{T}}$	$+X\mathbf{e}_c\mathbf{e}_l^{\mathrm{T}}$
$+XG$	$+X[dG/dZ]_{lc}$	$+X[dG/dX]_{lc} + \mathbf{e}_l\mathbf{e}_c^{\mathrm{T}}G$	
$+XG^{\mathrm{T}}$	$+X[dG/dZ]_{lc}^{\mathrm{T}}$	$+X^{\mathrm{T}}[dG/dX]_{lc}^{\mathrm{T}} + \mathbf{e}_l\mathbf{e}_c^{\mathrm{T}}G^{\mathrm{T}}$	

For the first few applications of (4.1) the derivatives will be sparse, and for small p, q and/or n it may be worthwhile to exploit this sparsity. There are 5 possible sparsity patterns for dF/dX:

1) all elements are zero
2) in the (i, j)-block only the (i, j)-element is nonzero
3) only the i-th row in the (i, j)-block is nonzero
4) only the j-th column in the (i, j)-block is nonzero
5) the matrix is full

As an example, let $p = 1$, $q = 2$ and consider the differentiation of C_0, C_1, and C_2. These matrices are given by $C_0 = \Sigma$, $C_1 = A_1\Sigma + B_1\Sigma$ (the first operation of (4.1) twice) and $C_2 = A_1C_1 + B_2\Sigma$ (the third operation of (4.1) followed by the first operation). Treating Σ as non-symmetric to begin with, one obtains:

$$\begin{array}{lll}
dC_0/dA_1 = 0 & [dC_1/dA_1]_{lc} = \mathbf{e}_l\mathbf{e}_c^{\mathrm{T}}\Sigma & [dC_2/dA_1]_{lc} = A_1[dC_1/dA_1]_{lc} + \mathbf{e}_l\mathbf{e}_c^{\mathrm{T}}C_2 \\
dC_0/dB_1 = 0 & [dC_1/dB_1]_{lc} = \mathbf{e}_l\mathbf{e}_c^{\mathrm{T}}\Sigma & [dC_2/dB_1]_{lc} = A_1[dC_1/dB_1]_{lc} + \mathbf{e}_l\mathbf{e}_c^{\mathrm{T}}C_2 \\
dC_0/dB_2 = 0 & dC_1/dB_2 = 0 & [dC_2/dB_2]_{lc} = \mathbf{e}_l\mathbf{e}_c^{\mathrm{T}}\Sigma \\
[dC_0/d\Sigma]_{lc} = \mathbf{e}_l\mathbf{e}_c^{\mathrm{T}} & [dC_1/d\Sigma]_{lc} = (A_1 + B_1)\mathbf{e}_l\mathbf{e}_c^{\mathrm{T}} & [dC_2/d\Sigma]_{lc} = A_1[dC_1/d\Sigma]_{lc} + B_2\mathbf{e}_l\mathbf{e}_c^{\mathrm{T}}
\end{array}$$

Here all the sparsity patterns are represented and the only full matrices are the derivatives of C_2 with respect to A_1 and B_1. Finally, the derivatives with respect to the symmetric Σ are adjusted using (A.7).

Now we turn attention to the vector-Yule-Walker equations (2.10). Differentiating through these with respect to a parameter gives:

$$\begin{aligned} S'_{j,lc} - \left(A_1 S'_{j-1,lc} + \ldots + A_j S'_{0,lc}\right) - \left(A_{j+1}(S'_{1,lc})^{\mathrm{T}} + \ldots + A_p(S'_{p-j,lc})^{\mathrm{T}}\right) \\ = G'_{lc} + \left(A'_{1,lc}S_{j-1} + \ldots + A'_{j,lc}S_0\right) + \left(A'_{j+1,lc}S_1^{\mathrm{T}} + \ldots + A'_p S_{p-j}^{\mathrm{T}}\right) \text{ for } j = 0,\ldots,p. \end{aligned} \tag{4.2}$$

This set of equations has exactly the same coefficient matrix as the original equations (2.10), but a different right hand side which can be obtained using the formula for $d(XA)/dX$ in (A.4) (sparsity can be exploited). It can therefore be solved to obtain the derivatives of $S_0,...,S_p$ using the same factorization as was used to obtain the S_j.

4.2 Remaining steps in likelihood gradient calculation

It follows from (4.1) that the derivative of $\mathbf{y}_t$ (and thereby $\mathbf{w}_t$) with respect the B_j's and Σ is zero, and (A.5) gives its derivative with respect to the A_j's and $\boldsymbol{\mu}$. For complete data, the next needed derivative is that of L, the Cholesky factor of Ω. As the derivative of all the submatrices of Ω have been found, this may be obtained using (A.10) and (A.11), making necessary modifications to take advantage of the block-band structure of Ω. To finish the calculation of the gradient of $l(\boldsymbol{\theta})$ in (2.13), use $L\mathbf{z}$ = $\mathbf{w}$ together with (A.8) to differentiate $\mathbf{z}$, followed by (A.6) to differentiate $\mathbf{z}^{\mathrm{T}}\mathbf{z}$, and finally use $d(\log l_{ii})/dX = (1/l_{ii})dl_{ii}/dX$.

In the missing value case, the operations that must be differentiated are the same: matrix products, Cholesky factorization, forward substitution, and determinants of lower triangular matrices, and there is no need to give details of all of them. They have been implemented in [Jonasson 200 xxx] by writing functions that implement (A.3), (A.9), (A.10) and (A.11).

4.3 Operation count for gradient calculation and possible savings

Inspection of the formulae in Appendix A for the derivatives of the most costly operations, namely matrix products, Cholesky factorization and forward substitution, shows that they all cost approximately $2n_\theta$ times more multiplications than the original operations being differentiated, where n_θ = $r^2(p+q) + r(r+1)/2$ is the total number of model parameters excluding $\boldsymbol{\mu}$ which does not enter the costly operations. The gradient calculation will therefore usually dominate the total work needed for likelihood maximization and this is confirmed by the numerical results of Section 5.

One way of trying to reduce this work would be to use numerical gradients in the beginning iterations, when the accuracy of the gradients is not as important as closer to the solution. Using forward differencing, $(\partial/\partial\theta_k)l(\theta)$ = $\left(l(\theta + \delta\mathbf{e}_k) - l(\theta)\right)/\delta$, the gradient can be approximated with n_θ function calls, giving a potential saving of factor 2. However, judging by the results shown in Table II in the next section, it seems that this technique is not so useful.

Another possibility of speeding the computations exists when estimating seasonal models, structured models, or various models with constraints on the parameters such as distributed lag models. Without entering too much into detail, such models may often be described by writing θ as a function of a reduced set of parameters, $\theta = g(\phi)$, where $\phi \in \mathbb{R}^{n_\phi}$ has (often much) fewer components than θ. The log-likelihood for a given set of parameters ϕ is $l(g(\phi))$, and the corresponding gradient is $l'(g(\phi))J_g(\phi)$, where J_g is the $n_\theta \times n_\phi$ Jacobian of the transformation g. The parameter matrices may be sparse and it would be possible to exploit the sparsity, but big savings are also possible by multiplying with the Jacobian earlier in the computation of the gradient, instead of after evaluating $l'(\theta)$. A convenient place to make the change of variables is after the differentiation of **w**, the C_j's, G_j's and W_j's, and the $\mathbf{g}_j$'s in the right-hand-side of (B.3). The costly derivatives come after this, so the potential saving approaches a factor of n_θ / n_ϕ. In [Jonasson 200 xxx] this course of action has been implemented, and the likelihood routines have J_g as an optional parameter.

5. NUMERICAL EXPERIMENTS

The methods described in Sections 2−4 have been implemented in Matlab as described in the companion paper [Jonasson 200 xxx]. The Matlab package also includes a function to simulate time series as described in Appendix C below, and a program demonstrating likelihood maximization. In this section we report on some experiments carried out with this package.

5.1 Likelihood maximization

The primary use of likelihood evaluation is to estimate model parameters by maximizing the likelihood function. The companion paper contains a demonstration program of such parameter estimation. Among the models demonstrated are VAR(2) with $r = 3$, $n = 400$ and complete data, VAR(2) with $r = 3$, $n = 200$ and 5% missing values (missing pattern "miss-5a" from Table I), and two VARMA(1, 1) with $r = 2$, $n = 200$, one with complete data and the other with 5% missing. Choosing the *ucminf* optimization routine (see the companion paper), the parameter estimates for these models were obtained using 34, 37, 31 and 47 function-gradient evaluations respectively. Using Matlab 7.1 with its default Intel Math Kernel Library (MKL) on an 1830 MHz L2500 Core Duo processor (Lenovo X60s computer) the total execution time for this modelling was 62 seconds. The *fminunc* optimizer takes about 60−70% longer. The demonstration program also estimates two VAR models with constraints on the parameters, of the type discussed near the end of Section 4.3. The modelling involves real meteorological data with $p = 2$ and 3, $r = 3$, $n = 146$ and 6.6% of the values missing. Details of the models and data are in the companion paper Jonasson [200 xxx] and the data is further described by Hanna, Jonsson and Box [2004]. The estimation of these two models takes 39 and 41 function and gradient evaluations in 8 and 11 seconds respectively.

More complicated models require more iterations and longer execution time. To take some examples, a VARMA(1, 1) fit with $r = 4$ and $n = 400$ took 97 function and gradient evaluations and 7:40 minutes of execution time and the two meteorological models with $n = 208$ took 54 and 55 function and gradient evaluations 2:36 and 3:09 minutes of execution time.

5.2 Timing of function evaluations

The simulation function has been used to generate test data with several models, missing value patterns and dimensions, and these data have been used to test and time the likelihood evaluation functions. The tests were run on a 1600 MHz Pentium M processor (about 3 times slower than the one used for Section 5.1). Table I shows the run time in seconds required for one function evaluation for each combination of model, missing value pattern, and dimensions.

For the pure VAR models the simplifications of Section 3.3 are realized, and for complete data the solution to the vector-Yule-Walker equations and the calculation of **w** and **z** will govern the computation. For VMA and VARMA models these calculations still make up a portion of the total, but the factorization of Ω is now more expensive and accounts for most of the difference between the complete data execution times of the VAR(1) and VMA(1) models shown in Table I.

Table I. Run time in seconds per one function evaluation. The missing value patterns shown in the “data” column are a) complete data; b) miss-5a: 5% missing scattered in first quarter of each series; c) miss-5b: 5% missing scattered throughout entire series; d) miss-25: 25% missing — half the series have the first half missing.

		Dimension $r = 2$		Dimension $r = 4$		Dimension $r = 8$	
Model	Data	$n = 100$	$n = 500$	$n = 100$	$n = 500$	$n = 100$	$n = 500$
VAR(1)	complete	0.01	0.02	0.01	0.03	0.01	0.03
	miss-5a	0.02	0.10	0.03	0.24	0.05	0.85
	miss-5b	0.03	0.24	0.04	0.58	0.10	2.21
	miss-25	0.04	0.73	0.08	4.07	0.35	28.17
VMA(1)	complete	0.04	0.19	0.04	0.21	0.05	0.24
	miss-5a	0.06	0.29	0.06	0.47	0.09	1.07
	miss-5b	0.07	0.43	0.08	0.86	0.15	2.78
	miss-25	0.08	1.00	0.12	4.41	0.39	28.34
VAR(3)	complete	0.01	0.03	0.01	0.03	0.04	0.06
	miss-5a	0.03	0.11	0.04	0.24	0.08	0.88
	miss-5b	0.03	0.19	0.05	0.55	0.12	2.19
	miss-25	0.04	0.73	0.09	4.09	0.37	27.90
VARMA(2,2)	complete	0.05	0.25	0.06	0.27	0.08	0.34
	miss-5a	0.07	0.33	0.08	0.51	0.13	1.24
	miss-5b	0.08	0.57	0.10	1.02	0.19	2.98
	miss-25	0.09	1.02	0.14	4.47	0.44	28.64

Missing values add gradually to the cost, and when there are few missing values the execution time is only marginally greater than for complete data. When more values are missing the savings in the VAR model are gradually eradicated. Now the approximately order M^3 operations (independent of p and q) involving the profile-sparse $N \times M$ matrices $\hat{V}$ and $\hat{\Lambda}_{\text{om}}$, and the full $M \times M$ matrices S_{m}, R_{Λ}, R_V, P, Q, R and K become more and more important. If these were the only computations one would expect a factor 125 difference between $n = 100$ and $n = 500$, but because of other calculations that do not depend on M the largest factor in the table is 80 (for VAR(1), miss-25, $r = 8$).

Another feature shown by the table is the difference between miss-5a and miss-5b, corroborating the discussion between equations (3.5) and (3.6) in Section 3.1. This ranges from a factor of 1.26 to a factor of 2.61, the average being 1.89.

5.3 Timing of gradient evaluations

Timing experiments for gradient evaluation were also carried out. It seems most relevant to compare with the cost of numerical differentiation. Therefore Table II shows the factor between the time of one gradient evaluation and m function evaluations. Where a table entry is less than one, the analytical gradients take less time than (maybe inaccurate) forward-difference numerical gradients, and where it is less than two the analytical gradients are cheaper than central-difference numerical gradients. The average of the 70 factors shown in the table is 0.74. The table is less extensive than Table I because the computer used did not have enough memory to time the largest models. The memory was sufficient to time some runs not shown in the table, and the results were comparable to the figures shown (the average factor for 11 cases not shown in the table was 0.41).

Table II. Execution time for one gradient evaluation divided by time for m function evaluations where m is the number of model parameters. See caption of Table I for explanation of the "Data" column.

		Dimension r = 2		Dimension r = 4		r = 8
Model	Data	n = 100	n = 500	n = 100	n = 500	n = 100
VAR(1)	complete	0.40	0.47	0.15	0.17	0.09
	miss-5a	0.56	0.78	0.35	0.97	0.66
	miss-5b	0.58	0.66	0.50	1.02	0.77
	miss-25	0.83	2.04	1.33	2.22	2.11
VMA(1)	complete	1.16	1.57	1.06	1.08	1.12
	miss-5a	0.63	0.68	0.33	0.66	0.52
	miss-5b	0.67	0.76	0.42	0.74	0.65
	miss-25	0.71	1.39	1.02	2.01	1.92
VAR(3)	complete	0.23	0.26	0.10	0.11	0.05
	miss-5a	0.35	0.62	0.30	1.09	0.50
	miss-5b	0.38	0.82	0.42	1.05	0.72
VARMA(2,2)	complete	1.10	1.19	1.07	1.17	1.08
	miss-5a	0.34	0.45	0.27	0.66	0.55
	miss-5b	0.36	0.51	0.38	0.75	0.74

The relative cost of gradient evaluation is somewhat lower than expected at the outset, as the derivative of many basic linear algebra operations with the formulae of Appendix A cost $2m$ times more than the operations themselves. This could be because the evaluation of the gradient involves larger matrices, thus making better use of the Intel MKL. The variable power of the MKL explains partly the variability of the numbers in Table II, but the rest of the disparity probably occurs because different derivative routines make unlike use of the power of Matlab.

APPENDICES

A. DIFFERENTIATION WITH RESPECT TO MATRICES

Many of the identities that follow may be found in [Nel 1980]; see also [Golub and Van Loan 1983]. If f is differentiable function on the set of $M \times N$ matrices, f: $\mathbb{R}^{M\times N} \to \mathbb{R}$, then the $N \times M$ matrix with (i, j)-element $\partial f/\partial x_{ij}$ will be denoted by $f'(X)$ or df/dX. If $\mathbf{f}$ is a vector valued function of a matrix, $\mathbf{f}$: $\mathbb{R}^{M\times N} \to \mathbb{R}^m$ then $d\mathbf{f}/dX$ or $\mathbf{f}'(X)$ denotes the block matrix:

$$\begin{bmatrix} \partial\mathbf{f}/\partial x_{11} & \cdots & \partial\mathbf{f}/\partial x_{1N} \\ \vdots & & \vdots \\ \partial\mathbf{f}/\partial x_{M1} & \cdots & \partial\mathbf{f}/\partial x_{MN} \end{bmatrix},$$

where each block is an m-dimensional column vector (the k-th block row is actually the Jacobian matrix of $\mathbf{f}$ with respect to the k-th row of X). If F is matrix valued, F: $\mathbb{R}^{M\times N} \to \mathbb{R}^{m\times n}$, then dF/dX or $F'(X)$ denotes the $M \times N$ block-matrix

$$\begin{bmatrix} \partial F/\partial x_{11} & \cdots & \partial F/\partial x_{1N} \\ \vdots & & \vdots \\ \partial F/\partial x_{M1} & \cdots & \partial F/\partial x_{MN} \end{bmatrix}. \tag{A.1}$$

The (l,c)-block of (A.1) will be denoted by F'_{lc} or $[dF/dX]_{lc}$ and it is an $m \times n$ matrix with (i, j)-element equal to $\partial f_{ij}(X)/\partial x_{lc}$. It is now easy to verify, that if a is a scalar and $\tilde{F}$ is another matrix function with same dimensions as F, then $d(aF + \tilde{F})/dX = a\,dF/dX + d\tilde{F}/dX$. We also have (where $\mathbf{e}_l$ is the l-th unit vector):

$$[dX/dX]_{lc} = \mathbf{e}_l\mathbf{e}_c^{\mathrm{T}}, \tag{A.2}$$

and, if G is another matrix function G: $\mathbb{R}^{M\times N} \to \mathbb{R}^{n\times k}$, then

$$[d(FG)/dX]_{lc} = FG'_{lc} + F'_{lc}G. \tag{A.3}$$

A.1 Differentiation of matrix products

The following special cases are all consequences of (A.2) and (A.3):

$$\begin{aligned}
&[dX^{\mathrm{T}}/dX]_{lc} = \mathbf{e}_c\mathbf{e}_l^{\mathrm{T}} & & \\
&[d(AX)/dX]_{lc} = A\mathbf{e}_l\mathbf{e}_c^{\mathrm{T}} & &[d(XA)/dX]_{lc} = \mathbf{e}_l\mathbf{e}_c^{\mathrm{T}}A \\
&[d(AF)/dX]_{lc} = AF'_{lc} & &[d(FA)/dX]_{lc} = F'_{lc}A \\
&[d(XF))/dX]_{lc} = XF'_{lc} + \mathbf{e}_l\mathbf{e}_c^{\mathrm{T}}F & &[d(FX)/dX]_{lc} = F'_{lc}X + F\mathbf{e}_l\mathbf{e}_c^{\mathrm{T}}
\end{aligned} \tag{A.4}$$

where, in each case, A is a constant matrix with dimensions compatible with those of F and X. When A is actually a vector, $A = \mathbf{a}$, we have:

$$d(X\mathbf{a})/dX = \begin{bmatrix} \mathbf{e}_1 \\ \vdots \\ \mathbf{e}_M \end{bmatrix} \mathbf{a}^{\mathrm{T}}, \tag{A.5}$$

and similarly, $d(\mathbf{a}^{\mathrm{T}}X)/dX = \mathbf{a}[\mathbf{e}_1^{\mathrm{T}} \ \cdots \ \mathbf{e}_N^{\mathrm{T}}]$. If $n = 1$ and F is vector-valued, $F = \mathbf{f} = [f_1, \ldots, f_m]^{\mathrm{T}}$, the l-th block-row of $d(X\mathbf{f})/dX$ is $X[\partial\mathbf{f}/\partial x_{l1} \ \ldots \ \partial\mathbf{f}/\partial x_{lN}] + \mathbf{e}_l\mathbf{f}^{\mathrm{T}}$ and the c-th block-column of $d(\mathbf{f}^{\mathrm{T}}X)/dX$ is $[\partial\mathbf{f}/\partial x_{1c} \ \ldots \ \partial\mathbf{f}/\partial x_{Mc}]^{\mathrm{T}}X + \mathbf{f}\mathbf{e}_c^{\mathrm{T}}$. Furthermore:

$$[d(\mathbf{f}^{\mathrm{T}}\mathbf{f})/dX]_{lc} = 2\mathbf{f}^{\mathrm{T}}(\partial\mathbf{f}/\partial x_{lc}) \tag{A.6}$$

A.2 Derivative with respect to a symmetric matrix

When X is square and symmetric and its upper triangle duplicates its lower triangle, the correct derivatives are obtained by using the full X in (A.4), and assigning in the final result:

$$(l,c)\text{-block} \leftarrow (l,c)\text{-block} + (c,l)\text{-block} \quad (\text{for all } l, c \text{ with } l > c) \tag{A.7}$$

(only the lower block-triangle is relevant). To take an example let $n = 2$, $x_{21} = x_{12}$ and consider the calculation of dX^2/dx_{21}. By (A.2) and (A.4),

$$dX^2/dx_{21} = X\mathbf{e}_2\mathbf{e}_1^{\mathrm{T}} + \mathbf{e}_2\mathbf{e}_1^{\mathrm{T}}X = \begin{bmatrix} x_{12} & 0 \\ x_{22}+x_{11} & x_{12} \end{bmatrix} \text{ and } dX^2/dx_{12} = X\mathbf{e}_1\mathbf{e}_2^{\mathrm{T}} + \mathbf{e}_1\mathbf{e}_2^{\mathrm{T}}X = \begin{bmatrix} x_{21} & x_{11}+x_{22} \\ 0 & x_{21} \end{bmatrix}.$$

Adding these matrices and letting x denote the duplicated element in X (i.e. $x = x_{12} = x_{21}$) gives the matrix:

$$\begin{bmatrix} 2x & x_{11}+x_{22} \\ x_{11}+x_{22} & 2x \end{bmatrix}$$

which is easily verified to be the derivative of X^2 with respect to x. It would be possible to make the calculation of derivatives with respect to symmetric matrices more efficient by developing appropriate formulae analogous to (A.4), but the complications would probably be significant and the pay-back marginal in the present setting.

A.3 Derivative of the solution to linear equations

If the vector $\mathbf{y}$ is given by $A\mathbf{y} = \mathbf{b}$ then it follows from (A.3) that $A(\partial\mathbf{y}/\partial x_{lc}) + A'_{lc}\mathbf{y} = \partial\mathbf{b}/\partial x_{lc}$ and $\partial\mathbf{y}/\partial x_{lc}$ is therefore given by solving the set of linear equations:

$$A(\partial \mathbf{y}/\partial x_{lc}) = \partial \mathbf{b}/\partial x_{lc} - A'_{lc}\mathbf{y}. \tag{A.8}$$

We note that the factorization of A used to obtain $\mathbf{y}$ can be reused to obtain its derivative. Similarly, if the matrix F is given by $AF = B$ then F'_{lc} may be obtained by solving:

$$AF'_{lc} = B'_{lc} - A'_{lc}F. \tag{A.9}$$

A.4 Derivative of Cholesky factorization

If $S = LL^{\mathrm{T}}$ is the Cholesky factorization of a symmetric matrix S it follows from (A.3) that $S'_{lc} = L(L'_{lc})^{\mathrm{T}} + L'_{lc}L^{\mathrm{T}}$. If S'_{lc}, L and L'_{lc} are partitioned as follows for a given k

$$S'_{lc} = \begin{bmatrix} S'_1 & & \\ \mathbf{s}'^{\mathrm{T}} & s'_{kk} & \\ S'_2 & \mathbf{t} & S_3 \end{bmatrix},\ L = \begin{bmatrix} L_1 & & \\ \mathbf{u}^{\mathrm{T}} & l_{kk} & \\ L_3 & \mathbf{v} & L_2 \end{bmatrix} \text{and } L'_{lc} = \begin{bmatrix} L'_1 & & \\ \mathbf{u}'^{\mathrm{T}} & l'_{kk} & \\ L'_3 & \mathbf{v}' & L'_2 \end{bmatrix}$$

then $2(\mathbf{u}^{\mathrm{T}}\mathbf{u}' + l_{kk}l'_{kk}) = s'_{kk}$ and $L_1\mathbf{u}' + L'_1\mathbf{u} = \mathbf{s}'$ so that

$$L_1\mathbf{u}' = \mathbf{s}' - L'_1\mathbf{u} \tag{A.10}$$

and

$$l'_{kk} = (s'_{kk}/2 - \mathbf{u}^{\mathrm{T}}\mathbf{u}')/l_{kk}. \tag{A.11}$$

These relations may be used iteratively for k = 1, 2, … to calculate L'_{lc} line by line, with $\mathbf{u}'$ obtained from (A.10) with forward substitution.

B. SOLUTION OF THE VECTOR YULE-WALKER EQUATIONS

In this appendix, we consider the solution to the system of equations (2.10). Our approach closely resembles that given by [Mauricio 1997, eq. (6)] and in particular the system we solve is of the same order, namely $r^2p - r(r-1)/2$. However, Mauricio does not provide a derivation of the system, our notation is significantly different from his, and lastly the system solved is not exactly the same (although it is equivalent). Therefore, we provide an explicit derivation in this appendix.

Isolating S_p in the last equation of (2.10) and substituting into the first equation gives

$$S_0 - (A_1^{\mathrm{T}}S_1^{\mathrm{T}} + \ldots + A_{p-1}S_{p-1}^{\mathrm{T}}) - (A_pS_0A_p^{\mathrm{T}} + A_pS_1^{\mathrm{T}}A_{p-1}^{\mathrm{T}} + \ldots + A_pS_{p-1}^{\mathrm{T}}A_1^{\mathrm{T}}) = G_0 + A_pG_p^{\mathrm{T}} \tag{B.1}$$

It is convenient to make use of the Kronecker product ($A \otimes B$ is a block matrix with (i, j)-block equal to $a_{ij}B$), the notation vec A for the vector consisting of all the columns of a matrix A placed one after another, and vech A for the columns of the lower triangle of A placed one after another. A useful property here is $\operatorname{vec}(ASB^{\mathrm{T}}) = (B \otimes A)\operatorname{vec}S$. Let $\mathbf{s}_i = \operatorname{vec}S_i$, $\mathbf{g}_i = \operatorname{vec}G_i$, and denote the k-th column of A_i with $\mathbf{a}_{ik}$ and the k-th unit vector with $\mathbf{e}_k$. Because S_0 is symmetric, taking the transpose of (B.1) gives with this notation:

$$(I - A_p \otimes A_p)\mathbf{s}_0 - (A_1 \otimes I + A_p \otimes A_{p-1})\mathbf{s}_1 - \ldots - (A_{p-1} \otimes I + A_p \otimes A_1)\mathbf{s}_p = \operatorname{vec}G_0^{\mathrm{T}} + (A_p \otimes I)\mathbf{g}_p \tag{B.2}$$

Furthermore, the equations with right hand side $G_1,...,G_{p-1}$ in (2.10) may be written as

$$\begin{aligned} \mathbf{s}_1 - \hat{A}_1\mathbf{s}_0 - \tilde{A}_2\mathbf{s}_1 - \ldots - \tilde{A}_p\mathbf{s}_{p-1} &= \mathbf{g}_1 \\ \mathbf{s}_2 - \hat{A}_1\mathbf{s}_1 - \hat{A}_2\mathbf{s}_0 - \tilde{A}_3\mathbf{s}_1 - \ldots - \tilde{A}_p\mathbf{s}_{p-2} &= \mathbf{g}_2 \\ &\vdots \\ \mathbf{s}_{p-1} - \hat{A}_1\mathbf{s}_{p-2} - \ldots - \hat{A}_{p-1}\mathbf{s}_0 - \tilde{A}_p\mathbf{s}_1 &= \mathbf{g}_{p-1} \end{aligned} \tag{B.3}$$

where $\hat{A}_i = I \otimes A_i$ and $\tilde{A}_i$ is also an $r^2 \times r^2$ sparse block-matrix (which cannot be represented using $\otimes$):

$$\tilde{A}_i = \begin{bmatrix} \mathbf{a}_{i1}\mathbf{e}_1^{\mathrm{T}} & \cdots & \mathbf{a}_{ir}\mathbf{e}_1^{\mathrm{T}} \\ \vdots & & \vdots \\ \mathbf{a}_{i1}\mathbf{e}_r^{\mathrm{T}} & \cdots & \mathbf{a}_{ir}\mathbf{e}_r^{\mathrm{T}} \end{bmatrix}.$$

Together (B.2) and (B.3) provide pr^2 linear equations in the elements of $\mathbf{s}_0,\ldots,\mathbf{s}_{p-1}$, but one can (and should) take into account that S_0 is symmetric and $\mathbf{s}_0$ contains therefore duplicated elements. Let $\hat{S}_0$ be a lower triangular matrix such that $S_0 = \hat{S}_0 + \hat{S}_0^{\mathrm{T}}$ (the diagonal elements of $\hat{S}_0$ are halved compared with S_0) and let $\hat{\mathbf{s}}_0$ = vech$\,S_0$ (the $(r(r+1)/2)$-vector obtained by removing the duplicated elements from $\mathbf{s}_0$). Let also J be an $r^2 \times r(r+1)/2$ matrix such that post-multiplication with it removes columns r + 1, 2r + 1, 2r + 2, 3r + 1, 3r + 2, 3r + 3, …, r^2-1. Then a term of the type $\hat{A}_i\mathbf{s}_0$ in (B.3) may be rewritten:

$$\hat{A}_i\mathbf{s}_0 = (I \otimes A_i)\mathbf{s}_0 = \operatorname{vec}(A_i S_0 I^{\mathrm{T}}) = \operatorname{vec}(A_i\hat{S}_0 + A_i\hat{S}_0^{\mathrm{T}}) = (\hat{A}_i + \tilde{A}_i)\operatorname{vec} S_0 = (\hat{A}_i + \tilde{A}_i)J\hat{\mathbf{s}}_0, \qquad \text{(B.4)}$$

For (B.2) it is not difficult to verify that

$$(A_p \otimes A_p)\mathbf{s}_0 = (A_p \otimes A_p + \tilde{A}_p\hat{A}_p)\operatorname{vec}(S_0) = (A_p \otimes A_p + \tilde{A}_p\hat{A}_p)J\hat{\mathbf{s}}_0,$$

and furthermore that $\mathbf{s}_0 = D\operatorname{vec}(\hat{S}_0)$ where D is diagonal with d_{ii} = 2 when i corresponds to a diagonal element in S_0 (i.e. i = 1, r + 2, 2r +3, …); otherwise d_{ii} = 1. The matrix $\tilde{A}_p\hat{A}_p$ is a block-matrix with (i, j)-block equal to $\mathbf{a}_{pj}\mathbf{a}_{pi}^{\mathrm{T}}$.

Finally, the upper triangle of (B.2) should be removed. These modifications result in $r^2p - r(r-1)/2$ equations in the same number of unknowns, the elements of $\hat{\mathbf{s}}_0$, $\mathbf{s}_1,\ldots,\ \mathbf{s}_p$; (B.2) becomes

$$J^{\mathrm{T}}\Big((D - A_p \otimes A_p - \tilde{A}_p\hat{A}_p)J\hat{\mathbf{s}}_0 - \textstyle\sum_{i=1}^{p-1}(A_i \otimes I + A_p \otimes A_{p-i})\mathbf{s}_i\Big) = J^{\mathrm{T}}\Big(\operatorname{vec} G_0^{\mathrm{T}} + (A_p \otimes I)\mathbf{g}_p\Big) \qquad \text{(B.5)}$$

and (B.3) is modified using (B.4).

C. TIME SERIES SIMULATION

Simulation of VARMA time series has many applications e.g. to create test data for modelling methods, analyze such methods, and forecast with fitted models. Given values of $\boldsymbol{\varepsilon}_t$, $\mathbf{x}_t$ for $t = 1,\ldots,h$ where h = max(p,q) one may draw $\boldsymbol{\varepsilon}_t$ from $N(0,\Sigma)$ for t = $h+1, h+2,\ldots$ and apply (2.2) and (2.1) to obtain simulated values of $\mathbf{x}_t$ for $t > h$. If the starting values are not given, one may start with any values, for example zeros, and, after simulating, discard an initial segment to avoid spin-up effects. This is for example done in the routine *arsim* of [Schneider and Neumaier 2001]. For processes with short memory, this procedure works well and the discarded segment need not be very long, but for processes that are nearly non-stationary it may take a long time before they reach their long-term qualities, it is difficult to decide the required length of the initial segment, and the initial extra simulations may be costly. These drawbacks may be avoided by drawing values to start the simulation from the correct distribution.

Let $\mathbf{x}'$ = $(\mathbf{x}_1^{\mathrm{T}},\ldots,\mathbf{x}_h^{\mathrm{T}})^{\mathrm{T}}$ have mean $\boldsymbol{\mu}'$ and covariance matrix S', $\boldsymbol{\varepsilon}'$ = $(\boldsymbol{\varepsilon}_1^{\mathrm{T}},\ldots,\boldsymbol{\varepsilon}_h^{\mathrm{T}})^{\mathrm{T}}$ have covariance matrix Σ', and let C' = cov$(\mathbf{x}', \boldsymbol{\varepsilon}')$. S', Σ' and C' are given with (2.7) and (2.8) and solution of the vector-Yule-Walker equations (2.10) applying (3.5) if necessary, and $\boldsymbol{\mu}'$ is the rh-vector $(\boldsymbol{\mu}^{\mathrm{T}},\ldots,\boldsymbol{\mu}^{\mathrm{T}})^{\mathrm{T}}$. Starting values for $\mathbf{x}'$ may be drawn from $\mathrm{N}(\boldsymbol{\mu}',\Sigma')$, and starting values for $\boldsymbol{\varepsilon}'$ (that are needed if there are moving average terms) may be drawn from the conditional distribution of $\boldsymbol{\varepsilon}' \,|\, \mathbf{x}'$, which is normal with expectation $C'^{\mathrm{T}}S'^{-1}(\mathbf{x}' - \boldsymbol{\mu}')$ and covariance matrix $\Sigma' - C'^{\mathrm{T}}S'^{-1}C'$. This conditional distribution may also be used to draw $\boldsymbol{\varepsilon}'$ when $\mathbf{x}_1,\ldots,\ \mathbf{x}_h$ are given and $\boldsymbol{\varepsilon}_1,\ldots,\ \boldsymbol{\varepsilon}_h$ are unknown, for example when forecasting with a moving average model. This procedure has been implemented in [Jonasson 200 xxx].

D. DETERMINANT OF A LOW RANK UPDATE

The economical evaluation of the determinant of the covariance matrix of the observations in the missing value case, described at the end of Section 3.1, is based on the following theorem. As we have been unable to locate a proof of this useful fact in the published literature, we include it here for completeness. An immediate consequence of the theorem is that the determinant of a low rank update of an arbitrary matrix M may often be evaluated efficiently using $\det(M + UV^{\mathrm{T}}) = \det M \det(I + V^{\mathrm{T}} M^{-1} U)$, in this way complementing the Sherman-Morrison-Woodbury formula.

Theorem. If A is $m \times n$, B is $n \times m$ and I_m and I_n are the m-th and n-th order identity matrices then $\det(I_m + AB) = \det(I_n + BA)$.

Proof. Let C and D be $m \times m$ and $n \times n$ invertible matrices such that $CAD = \begin{pmatrix} I_k & 0 \\ 0 & 0 \end{pmatrix}$ and let $D^{-1}BC^{-1} = \begin{pmatrix} B_1 & B_2 \\ B_3 & B_4 \end{pmatrix}$ be a partitioning with B_1 a $k \times k$ matrix. Then

$$\begin{aligned}
\det(I_m + AB) &= \det\left(C^{-1}C + C^{-1}\begin{bmatrix} I_k & 0 \\ 0 & 0 \end{bmatrix} D^{-1}D \begin{bmatrix} B_1 & B_2 \\ B_3 & B_4 \end{bmatrix} C \right) \\
&= \det(C^{-1}) \det\left(I_m + \begin{bmatrix} I_k & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} B_1 & B_2 \\ B_3 & B_4 \end{bmatrix} \right) \det C \\
&= \det\begin{bmatrix} I_k + B_1 & B_2 \\ 0 & I_{m-k} \end{bmatrix} = \det(I_k + B_1) = \det\begin{bmatrix} I_k + B_1 & 0 \\ B_3 & I_{n-k} \end{bmatrix} \\
&= \det\left(I_n + D\begin{bmatrix} B_1 & B_2 \\ B_3 & B_4 \end{bmatrix} CC^{-1}\begin{bmatrix} I_k & 0 \\ 0 & 0 \end{bmatrix} D^{-1} \right) \\
&= \det(I_n + BA).
\end{aligned}$$

The matrices C and D may, for example, be obtained from the singular value decomposition of A [Golub and Van Loan 1983].

ACKNOWLEDGEMENT

We wish to thank Jón Kr. Arason for help with proving the theorem of Appendix D.

REFERENCES

ANSLEY, C. F. 1979. An algorithm for the exact likelihood of a mixed autoregressive-moving average process. *Biometrika* 66, 1, 59–65.

ANSLEY, C. F. AND KOHN, R. 1983. Exact likelihood of vector autoregressive-moving average process with missing or aggregated data. *Biometrika 70*, 1, 275−278.

BROCKWELL P. J. AND DAVIS, R. A. 1987. *Time Series: Theory and Methods*. Springer-Verlag, New York.

GOLUB, G. H. AND VAN LOAN, C. F. 1983. *Matrix Computations*. North Oxford Academic, Oxford.

HANNA, E., JONSSON, T. AND BOX, J. E. 2004. An analysis of Icelandic climate since the nineteenth century. *International Journal of Climatology 24*, 10, 1193-1210.

HARVEY, A. C. AND PHILLIPS, G. D. A. 1979. Maximum likelihood estimation of regression models with autoregressive-moving average disturbances. *Biometrika* 66, 1, 49–58.

JONASSON, K. 200 xxx, Algorithm xxx: Exact VARMA likelihood and its gradient for complete and incomplete data with Matlab. *ACM Trans. Math Softw. xxx*, xxx, xxx–xxx.

JONES, R. H. 1980, Maximum likelihood fitting of ARMA models to time series with missing observations. *Technometrics 22*, 3, 389–395.

LJUNG, G. M. 1989. A note on the estimation of missing values in time series. *Communic. Statist. – Simul. Comput. 18*, 2, 459–465.

LJUNG, G. M. AND BOX, G. E. P. 1979, The likelihood function of stationary autoregressive-moving average models, *Biometrika 66*, 2, 265–270.

LUCEÑO, A. 1994. A fast algorithm for the exact likelihood of stationary and nonstationary vector autoregressive-moving average processes. *Biometrika 81*, 3, 555–565.

MAURICIO, J. A. 1997. Algorithm AS 311: The exact likelihood function of a vector autoregressive moving average model. *Appl. Statist.46*, 1, 157–171.

MAURICIO, J. A. 2002. An algorithm for the exact likelihood of a stationary vector autoregressive moving average model. *J Time Series Analysis 23*, 4, 473–486.

MÉLARD, G., ROY, R. AND SAIDI, A. 2006. Exact maximum likelihood estimation of structured or unit root multivariate time series models. *Comput. Statist. Data Anal. 50*, 2957−2986.

METAXOGLOU, K. AND SMITH, A. 2007. Maximum likelihood estimation of VARMA models using a state space EM algorithm. *J Time Series Analysis*, forthcoming.

NEL, D. G. 1980. On matrix differentiation in statistics. *South African Statistical J. 15*, 2, 137–193.

NEUMAIER, A. AND SCHNEIDER, T. 2001. Estimation of parameters and eigenmodes of multivariate autoregressive models. *ACM Trans. Math. Softw. 27* 1, 27–57.

PENZER, J. AND SHEA, B. L. 1997. The exact likelihood of an autoregressive-moving average model with incomplete data. *Biometrika 84*, 4, 919–928.

PHADKE, M. S. AND KEDEM, G. 1978. Computation of the exact likelihood function of multivariate moving average models. *Biometrika 65*, 3, 511–19.

SCHNEIDER, T. AND NEUMAIER, A. 2001. Algorithm 808: ARfit - A Matlab package for the estimation of parameters and eigenmodes of multivariate autoregressive models. ACM Trans. Math. Softw. 27 1, 58–65.

SHEA, B. L. 1989. Algorithm AS 242: The exact likelihood of a vector autoregressive moving average model. *Appl. Statist. 38*, 1, 161–204.

SHERMAN, J. AND MORRISON, W. J. 1950. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix, *Ann. Math. Statist. 21*, 124–127.

SIDDIQUI, M. M. 1958. On the inversion of the sample covariance matrix in a stationary autoregressive process. *Ann. Math. Statist. 29*, 585–588.

WOODBURY, M. A. 1950. Inverting modified matrices, Memorandum Report 42, Statistical Research Group, Princeton University, Princeton NJ.