Title: Evaluating exact VARMA likelihood and its gradient when data are incomplete.

Authors: S. Ferrando, K. Jonasson.
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.


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 200x] 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 and Mauricio [2002] gives details of a complete data multivariate implementation of the Cholesky decomposition method.

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 by Neumaier and Schneider [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 pa-
per 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 \( x_t \in \mathbb{R}^r \) for integer \( t \) is given by:

\[
\begin{align*}
x_t - \mu &= \sum_{j=1}^{p} A_j (x_{t-j} - \mu) + y_t, \\
y_t &= \epsilon_t + \sum_{j=1}^{q} B_j \epsilon_{t-j},
\end{align*}
\]

where, \( \mu \) is the expected value of \( x_t \), the \( A_j \)'s and the \( B_j \)'s are \( r \times r \) matrices, and the \( \epsilon_t \)'s are \( r \)-variate \( N(0, \Sigma) \) uncorrelated in time. Let \( \theta \) 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, \( \Sigma \) and \( \mu; \Sigma \) being symmetric). If there are no missing values, observations \( x_t \) for \( t = 1, \ldots, n \) are given, and \( x \) denotes the \( nr \)-vector \((x_1^T, \ldots, x_n^T)^T\) of all these values. When there are missing values the observations are limited to a subvector \( x_o \in \mathbb{R}^N \) of \( x \), and \( x_m \in \mathbb{R}^M \) is a vector of the missing values, say \( x_m = (x_{m_1}, \ldots, x_{m_M}) \). If the time series is stationary then the complete data log-likelihood function is given by

\[
l(\theta) = -\frac{1}{2} \left( nr \log 2\pi + \log \det S + (x - \mu)^T S^{-1} (x - \mu) \right)
\]

where \( S = \text{cov}_\theta(x) \) and \( \mu = \text{E}_\theta(x) = (\mu^T, \ldots, \mu^T)^T \). The log-likelihood function for the observed data is given by
\[
I_0(\theta) = -\frac{1}{2} \left( N \log 2\pi + \log \det S_{\theta} + (x_\theta - \bar{\mu}_\theta)^T S_{\theta}^{-1} (x_\theta - \bar{\mu}_\theta) \right)
\]  
(2.4)

where \( S_{\theta} = \text{cov}_\theta(x_\theta) \) is obtained from \( S \) by removing rows \( m_1, \ldots, m_M \) and columns \( m_1, \ldots, m_M \) and \( \bar{\mu}_\theta = \mathbb{E}_\theta(x_\theta) \) is obtained from \( \bar{\mu} \) 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),

\[
y_t = x_t - \mu - \sum_{j=1}^p A_j (x_{t-j} - \mu)
\]

for \( t > p \). Let \( w_t = x_t - \mu \) for \( t \leq p \) and \( w_t = y_t \) for \( t > p \) and let \( w = (w_1^T, \ldots, w_p^T)^T \). Then \( w = \Lambda(x - \bar{\mu}) \) where \( \Lambda \) is the \( nr \times nr \) lower triangular block-band matrix given by

\[
\Lambda = \begin{bmatrix}
I & & \\
& \ddots & \\
& & I
\end{bmatrix}
- A_j \begin{bmatrix}
\ddots & \\
& -A_j & I
\end{bmatrix}
- A_j \begin{bmatrix}
\ddots & \\
& -A_j & I
\end{bmatrix}
\]

(2.5)

Now let \( c_j = \text{cov}(x_t, x_{t-j}), g_j = \text{cov}(y_t, x_{t-j}), w_j = \text{cov}(y_t, y_{t-j}) \) and \( s_j = \text{cov}(x_t, x_{t-j}) \), (all these are \( r \times r \) matrices). Note that with this notation,

\[
S = \begin{bmatrix}
s_0 & s_1^T & \cdots & s_{p-1}^T \\
s_1 & s_0 & \cdots & s_{p-2}^T \\
\vdots & \ddots & \ddots & \vdots \\
s_{p-1} & \cdots & s_1^T & s_0
\end{bmatrix}
\]

(2.6)

Furthermore, let \( A_j \) 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 \( e_{t-j}^T \) for \( j = 0, \ldots, q \) and taking expectations the following recurrence formulae for \( C_0, C_1, C_2, \ldots \) are obtained:

\[
C_j = A_j C_{j-1} + \cdots + A_1 C_0 + B_j \Sigma, \quad \text{for } j = 0, 1, \ldots
\]

(2.7)

(so \( C_0 = \Sigma \)). With \( B_0 = I \), we have by (2.1) and (2.2):
\[ G_j = B_j C_0^T + \ldots + B_j C_{q-j}^T, \text{ for } j = 0, \ldots, q, \quad (2.8) \]

\[ W_j = B_j \Sigma B_0^T + \ldots + B_j \Sigma B_{q-j}^T, \text{ for } j = 0, \ldots, q. \quad (2.9) \]

For \( j < 0 \) or \( j > q \), \( C_j, G_j \) and \( W_j \) are zero. By multiplying (2.1) from the right with \((T_j - \mu)\) 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):

\[ S_0 - A_0 S_1^T - \ldots - A_p S_p^T = G_0 \]
\[ S_1 - A_0 S_0 - A_1 S_1^T - \ldots - A_p S_{p-1}^T = G_1 \]
\[ S_2 - A_0 S_1 - A_2 S_0 - A_3 S_1^T - \ldots - A_p S_{p-2}^T = G_2 \]
\[ \vdots \]
\[ S_p - A_0 S_{p-1} - A_2 S_{p-2} - \ldots - A_p S_0 = G_p \quad (2.10) \]

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

\[
\Omega = \begin{bmatrix}
S_0 & S_1^T & \cdots & S_{p-1}^T \\
S_1 & S_0 & \cdots & \vdots \\
\vdots & \ddots & \ddots & \vdots \\
S_{p-1} & \cdots & S_1 & S_0 \\
G_q & \cdots & G_1 & W_0 \\
\vdots & \ddots & \ddots & \vdots \\
G_1^T & G_2^T & \cdots & G_q^T \\
W_0^T & \cdots & \cdots & W_q^T \\
W_1^T & \cdots & \cdots & W_q^T \\
W_q & \cdots & \cdots & W_q \\
\end{bmatrix}
\]

(2.11)

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

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

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

\[
l(\theta) = -\frac{1}{2} \left( nr \log 2\pi + \log \det \Omega + \mathbf{w}^T \Omega^{-1} \mathbf{w} \right) \quad (2.12)\]

To evaluate (2.12) it is most economical to calculate the Cholesky-factorization \( \Omega = LL^T \) 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

\[
l(\theta) = -\frac{1}{2} (nr \log 2\pi + 2\sum_i \log l_i + \mathbf{z}^T \mathbf{z}). \quad (2.13)\]
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 \( x \) it takes \( r^2 p(n - p) \) multiplications to calculate \( w \). Determining the \( C_j \)'s for \( j \leq q \), \( G_j \)'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 \( r \frac{n}{p} \) multiplications. The cost of the Cholesky-factorization of \( \Omega \) will be about \( 3 \left( \frac{h^2}{2} + \frac{(p/2 + q)(n - h)}{n} \right) \). 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 \( 8000 \frac{n}{6} \approx 8.5 \times 10^{9} \) multiplications (and take about 7 min. on a typical Intel computer) but calculation with (2.13), including all the steps leading to it, will take \( 4.0 \times 10^6 \) 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 \( (x_o - \bar{\mu})^T S_o^{-1} (x_o - \bar{\mu}) \). Let \( \tilde{\Omega}, \tilde{\Lambda} \) and \( \tilde{S} \) be obtained from \( \Omega, \Lambda \) and \( S \) by placing rows and columns \( m_1, \ldots, m_M \) after the other rows and columns and partition them as follows (with \( \Omega_o, \Lambda_o \) and \( S_o \) being \( N \times N \), and \( \Omega_m, \Lambda_m \) and \( S_m \) being \( M \times M \)):

\[
\tilde{\Omega} = \begin{bmatrix} \Omega_o & \Omega_{om} \\ \Omega_{mo} & \Omega_m \end{bmatrix}, \quad \tilde{\Lambda} = \begin{bmatrix} \Lambda_o & \Lambda_{om} \\ \Lambda_{mo} & \Lambda_m \end{bmatrix}, \quad \text{and} \quad \tilde{S} = \begin{bmatrix} S_o & S_{om} \\ S_{mo} & S_m \end{bmatrix}.
\]

By the definition of \( w, \Omega = \Lambda S \Lambda^T \) and therefore

\[
\Omega_o = \Lambda_o S_o \Lambda_o^T + \Lambda_o S_{om} \Lambda_{om}^T + \Lambda_{om} S_{mo} \Lambda_o^T + \Lambda_{om} S_m \Lambda_{om}^T.
\]

(3.1)

\( \Lambda_o \) is obtained from \( \Lambda \) by removing rows and corresponding columns, and it is therefore an invertible lower band matrix with unit diagonal and bandwidth at most \( rp \), and \( \Omega_o \) is obtained from \( \Omega \) 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

\[
\tilde{\Omega}_o = \Lambda_o S_o \Lambda_o^T
\]

(3.2)

and \( \tilde{w}_o = \Lambda_o (x_o - \bar{\mu}) \) we have \( (x_o - \bar{\mu})^T S_o^{-1} (x_o - \bar{\mu}) = \tilde{w}_o^T \tilde{\Omega}_o^{-1} \tilde{w}_o \). Also, from (3.1) and (3.2)
\[ \hat{\Omega}_o = \Omega_o - \Lambda_o S_{om}\Lambda_{om}^T - \Lambda_{om} S_{om}^T \Lambda_o^T - \Lambda_{om} S_{om} \Lambda_{om}^T \]  
(3.3)

(keep in mind that \( S \) is symmetric). The matrices \( S_{om}, \Lambda_{om} \) and \( S_o, \Lambda_{om} \) 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 \( \Omega_o \). 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:

\[ \hat{\Omega}_o = \Omega_o + U S_n^{-1} V^T - V S_m^{-1} U^T \]  
(3.4)

where \( U = \Lambda_o S_{om} \) and \( V = \Lambda_o S_{om} + \Lambda_{om} S_{om} \). 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 \( \Lambda S \). From (2.5), (2.6) and proceeding as when deriving (2.10) the following block representation of \( \Lambda S \) for the case \( q > p \) is obtained:

\[
\begin{bmatrix}
S_0 & S_1^T & \cdots & S_{n-1}^T \\
\vdots & \ddots & \ddots & \vdots \\
S_{p-1} & \cdots & S_0 & S_1^T & \cdots & S_{n-1}^T \\
G_p & \cdots & G_1 & G_0 & G_{n+p+1} \\
G_q & \cdots & G_1 & G_0 \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots \\
G_q & \cdots & G_0 
\end{bmatrix}
\]

For \( q \leq p \) the upper partition is the same but the lower partition is:

\[
\begin{bmatrix}
G_q & \cdots & G_1 & G_0 & G_{n+p+1} \\
G_q & \cdots & G_0 \\
\vdots & \ddots & \ddots & \ddots & \ddots \\
G_q & \cdots & G_0
\end{bmatrix}
\]

For \( S_{p+1}, \ldots, S_{n-1} \), multiply (2.1) from the right with \((x_{r-j} - \mu)^T\) for \( j = p+1, \ldots, n-1 \) and take expectations (as when deriving (2.10)), giving

\[ S_j = A_j S_{j-1} + A_2 S_{j-2} + \ldots + A_p S_{j-p} + G_j \]  
(3.5)

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

\[ G_{-j} = C_{-j}^T + B_{j+1} C_{j+1}^T + \ldots + B_{j+q} C_{j+q}^T \]

where the \( C_j \)'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 \( \Lambda S \) are zero, giving almost 50% sparsity. In practice the missing values will often occur near the beginning of the observation period.
and this implies that \( V \) will be sparser still. To take an example, if \( q = 1 \), \( r = 2 \), \( n = 6 \) and \( m = (2, 3, 4, 5, 9) \) then the sparsity pattern of \( V \) will be:

\[
\begin{bmatrix}
* & * & * & * & *
* & * & * & * & *
* & * & * & * & *
* & * & * & * & *
* & * & * & * & *
\end{bmatrix}
\]

The SMW formula applied to (3.4) gives

\[
\hat{\Omega}^{-1} = \hat{\Omega}_o^{-1} + \hat{\Omega}_o^{-1}VQ^{-1}V^T\hat{\Omega}_o^{-1}
\]

where

\[
\hat{\Omega}_o = \Omega_o + US_m^{-1}U^T
\]

(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):

\[
\hat{\Omega}_o^{-1} = \Omega_o^{-1} - \Omega_o^{-1}UR^{-1}U^T\Omega_o^{-1}
\]

If \( L_\hat{\Omega} \) is the Cholesky factor of \( R \) and \( K = L_\hat{\Omega}^{-1}U^T\Omega_o^{-1}V \) it follows that \( Q = S_m - V^T\hat{\Omega}_o^{-1}V + K^T K \). The first method that springs to mind to evaluate \( V^T\hat{\Omega}_o^{-1}V \) efficiently is to Cholesky factorize \( \Omega_o = LL^\top \), use forward substitution to obtain \( \hat{V} = L^\top V \) and form \( \hat{V}^\top \hat{V} \). However, with this procedure \( \hat{V} \) will be full and the computation of \( \hat{V}^\top \hat{V} \) will cost \( NM(M+1)/2 \) multiplications. In contrast, if an \( \hat{L}^\top L \)-factorization of \( \Omega_o \) is employed instead of Cholesky 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: \( \hat{L}^\top L \)-factorize \( \Omega_o = L_\hat{\Omega}^\top L_\hat{\Omega} \), and back-substitute to get \( \hat{V} = L_\hat{\Omega}^\top V \) and \( \hat{\Lambda}_o = L_\hat{\Omega}^\top \Lambda_o \), making use of known sparsity for all calculations (the sparsity structure of \( \hat{\Lambda}_o \) will be similar to that of \( \hat{V} \)). With \( R_v = \hat{V}^\top \hat{V} \), \( \hat{\Lambda}_o = \hat{\Lambda}_o^\top \hat{\Lambda}_o \) and \( P = \hat{\Lambda}_o^\top \hat{\Lambda}_o \) (again exploiting sparsity) we find that \( R = S_m + R_v + S_m R \hat{\Lambda}_o S_m - S_m P - P^T S_m \) (all matrices in this identity are full \( M \times M \)), \( K = L^\top K \) and \( Q = S_m - R_v + K^T K \). Let further \( L_\hat{Q} \) be the Cholesky factor of \( Q \). \( \hat{w}_o = L_\hat{Q}^\top \hat{w}_o \), \( \mathbf{u} = L_\hat{Q}^{-1}(\hat{V}^\top \hat{w}_o - S_m \hat{\Lambda}_o \hat{w}_o) \) and \( \mathbf{v} = L_\hat{Q}^{-1}(\hat{V}^\top \hat{w}_o - K^T \mathbf{u}) \). A little calculation then gives:

\[
(x_o - \hat{\mu}_o)^T S_m^{-1}(x_o - \hat{\mu}_o) = \hat{w}_o^T \hat{w}_o - \mathbf{u}^T \mathbf{u} + \mathbf{v}^T \mathbf{v}.
\]

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 \( \det(X \pm AY^{-1}A^\top) = \det X \det Y^{-1} \det(X \mp A^\top Y^{-1} A) \). From (3.3), (3.6) and the definition of \( L_\hat{Q} \), \( \det \hat{\Omega}_o = \det(\hat{\Omega}_o - VS_m^\top V^\top) = \det \hat{\Omega}_o \det S_m^{-1} \det(S_m - V^\top \hat{\Omega}_o^{-1} V) = \det \hat{\Omega}_o \det S_m^{-1} (L_\hat{Q})^2 \).

Similarly, \( \det \hat{\Omega}_o = \det(\hat{\Omega}_o + US_m^{-1}U^\top) = \det \hat{\Omega}_o \det S_m^{-1} \det(S_m + U^\top \hat{\Omega}_o^{-1} U) = \det \hat{\Omega}_o \det S_m^{-1} (L_\hat{Q}^{-1})^2 \).

Since \( \det \Lambda_o = 1 \) it now follows from (3.2) and the definition of \( L_\hat{Q} \) that

\[
\log \det S_o = 2 (\log \det L_o + \log \det L_\hat{\Omega} + \log \det L_\hat{Q} - \log \det S_m)
\]
3.2 Estimating missing values and shocks

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

$$\mathbf{x}_m^E = S_{mo} S_o^{-1} (\mathbf{x}_o - \overline{\mathbf{m}}_o) + \overline{\mathbf{m}}_m,$$

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

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

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

$$
\mathbf{x}_m^E = S_m \mathbf{v}_2 + \overline{\mathbf{m}}_m,
$$

and

$$
\mathbf{\varepsilon}^E = \tilde{C} \Lambda_0^T \varepsilon_0^T (\hat{\mathbf{w}}_u + \hat{V} (\mathbf{v}_1 - \mathbf{v}_2) - \hat{\Lambda}_m S_m \mathbf{v}_2)
$$

where $\mathbf{v}_1 = L_0^T \mathbf{v}$ and $\mathbf{v}_2 = L_\mathbf{R}^T (\mathbf{u} + K \mathbf{v}_1)$.

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 $\mathbf{y}_t = \mathbf{\varepsilon}_t$ 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 $\Sigma$. The upper left $S$-partition of $\Omega$ 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 $\Sigma$. For the missing value case, $\Omega_o$ needs to be Cholesky factorized. It is obtained by removing rows and corresponding columns from $\Omega$, 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:

$$
\begin{bmatrix}
\Sigma_{o1} & \Sigma_{o2} & \cdots & \Sigma_{o,n-p}
\end{bmatrix}.
$$
where $\Sigma_o$ contains rows and columns of $\Sigma$ corresponding to the observed indices at time $p+i$. To obtain $L_o$ it is therefore sufficient to Cholesky factorize $\Sigma_o$ for each missing pattern that occurs, which in all realistic cases will be much cheaper than Cholesky factorizing the entire $\Omega_o$-matrix.

### 3.4 Operation count for missing value likelihood

Finding the $C_j$’s, $G_j$’s and $W_j$’s will be identical to the complete data case. The Cholesky factorization of $\Sigma_o$ costs at most $r^2N(q^2/2 + 7/6)$ multiplications (unless the upper left partition is unusually big). Forming $A S$ costs about $r^3(2p + q)(n - p)$ multiplications. The cost of forming $\hat{\Lambda}_{om}$ 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 typically the missing values will be concentrated near the beginning (the series represent measurements that did not all start at the same time) and the cost will be much smaller. The cost of $R_v$, $R_\Lambda$ and $P$ also depends on the missing value pattern. In the worst case the symmetric $R_v$ and $R_\Lambda$ 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: $S_m P$ 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 $\det S_m$ 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 $\Omega$ will usually be negligible, and much cheaper than when $q > 0$. When nothing is missing, it is the number $rpN$ of multiplications to find $w$ and the number $rpN/2$ of multiplications of the forward substitution for $z$ 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 $\Omega$, 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 ap-
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 $r \times r$ 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 $\Sigma$), and it is not difficult to verify that they can all be obtained by applying a sequence of operations of the following types:

\begin{align*}
F &\leftarrow F + XY \\
F &\leftarrow F + XY^T \\
F &\leftarrow F + XG \\
F &\leftarrow F + XG^T
\end{align*}

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:

<table>
<thead>
<tr>
<th>Change to $F$:</th>
<th>$[dF/dZ]_{ic}$</th>
<th>$[dF/dX]_{ic}$</th>
<th>$[dF/dY]_{ic}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$+XY$</td>
<td>0</td>
<td>$+e_ie^TY$</td>
<td>$+Xe_ie^T$</td>
</tr>
<tr>
<td>$+XY^T$</td>
<td>0</td>
<td>$+e_ie^TY^T$</td>
<td>$+Xe_ie^T$</td>
</tr>
<tr>
<td>$+XG$</td>
<td>$+X[dG/dZ]_{ic}$</td>
<td>$+X[dG/dX]_{ic} + e_ie^TG$</td>
<td></td>
</tr>
<tr>
<td>$+XG^T$</td>
<td>$+X[dG/dZ]_{ic}$</td>
<td>$+X^T[dG/dX]_{ic} + e_ie^TG^T$</td>
<td></td>
</tr>
</tbody>
</table>

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 $\Sigma$ as non-symmetric to begin with, one obtains:
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 $\Sigma$ 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:

$$S'_{j,c} = (A'_{j,c} S'_{j-1,c} + \ldots + A'_{j,c} S'_{0,c}) - (A'_{j,c} (S'_{j,c})^T + \ldots + A'_{p,c} (S'_{p-j,c})^T) + G'_{j,c} + (A'_{j,c} S'_{j-1,c} + \ldots + A'_{j,c} S'_{0,c}) + (A'_{j,c} S'_{j-1,c}^T + \ldots + A'_{p,c} S'_{p-j,c})$$

for $j = 0, \ldots, p$. (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 $S_0, \ldots, S_p$ 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 $y_t$ (and thereby $w_t$) with respect the $B_j$'s and $\Sigma$ is zero, and (A.5) gives its derivative with respect to the $A_j$'s and $\mu$. For complete data, the next needed derivative is that of $L$, the Cholesky factor of $\Omega$. As the derivative of all the submatrices of $\Omega$ 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 $\Omega$. To finish the calculation of the gradient of $l(\theta)$ in (2.13), use $Lz = w$ together with (A.8) to differentiate $z$, followed by (A.6) to differentiate $z^Tz$, and finally use $d(\log l_t)/dX = (1/l_t)dl_t/dX$.

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 200x] 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 $n_\theta = r^2(p + q) + r(r + 1)/2$ is the total number of model parameters excluding $\mu$ 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, \((\partial l/\partial \theta)(\theta) = (l(\theta + \delta \theta) - l(\theta))/\delta\), the gradient can be approximated with \(n_g\) 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, structural 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 \(\theta\) as a function of a reduced set of parameters, \(\phi = g(\theta)\), where \(\phi \in \mathbb{R}^p\) has (often much) fewer components than \(\theta\). The log-likelihood for a given set of parameters \(\phi\) 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 \(g_j\)'s in the right-hand-side of (B.3). The costly derivatives come after this, so the potential saving approaches a factor of \(n_\theta/n_\phi\). In [Jonasson 200x] 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 200x]. 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 constrained VAR models (cf. section 4.3) of real (meteorological) data, with \(p = 2\) and 3, \(r = 3\) and \(n = 146\), and this takes 39 and 41 f-g evaluations in 8 and 11 seconds.
More complicated models require more iterations and more execution time. To take some examples, a VARMA(1, 1) fit with \( r = 4 \) and \( n = 400 \) took 97 f-g evaluations and 7:40 minutes of execution time and the two meteorological models with \( n = 208 \) took 54 and 55 f-g 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 \( \Omega \) 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.

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

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 $\widehat{\Lambda}_c$, and the full $M \times M$ matrices $S_m$, $R_A$, $R_V$, $P$, $Q$, $R$ 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>
<thead>
<tr>
<th>Model</th>
<th>Data</th>
<th>Dimension $r = 2$</th>
<th>Dimension $r = 4$</th>
<th>$r = 8$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$n = 100$ $n = 500$</td>
<td>$n = 100$ $n = 500$</td>
<td>$n = 100$</td>
</tr>
<tr>
<td>VAR(1)</td>
<td>complete</td>
<td>0.40 0.47</td>
<td>0.15 0.17</td>
<td>0.09</td>
</tr>
<tr>
<td></td>
<td>miss-5a</td>
<td>0.56 0.78</td>
<td>0.35 0.97</td>
<td>0.66</td>
</tr>
<tr>
<td></td>
<td>miss-5b</td>
<td>0.58 0.66</td>
<td>0.50 1.02</td>
<td>0.77</td>
</tr>
<tr>
<td></td>
<td>miss-25</td>
<td>0.83 2.04</td>
<td>1.33 2.22</td>
<td>2.11</td>
</tr>
<tr>
<td>VMA(1)</td>
<td>complete</td>
<td>1.16 1.57</td>
<td>1.06 1.08</td>
<td>1.12</td>
</tr>
<tr>
<td></td>
<td>miss-5a</td>
<td>0.63 0.68</td>
<td>0.33 0.66</td>
<td>0.52</td>
</tr>
<tr>
<td></td>
<td>miss-5b</td>
<td>0.67 0.76</td>
<td>0.42 0.74</td>
<td>0.65</td>
</tr>
<tr>
<td></td>
<td>miss-25</td>
<td>0.71 1.39</td>
<td>1.02 2.01</td>
<td>1.92</td>
</tr>
<tr>
<td>VAR(3)</td>
<td>complete</td>
<td>0.23 0.26</td>
<td>0.10 0.11</td>
<td>0.05</td>
</tr>
<tr>
<td></td>
<td>miss-5a</td>
<td>0.35 0.62</td>
<td>0.30 1.09</td>
<td>0.50</td>
</tr>
<tr>
<td></td>
<td>miss-5b</td>
<td>0.38 0.82</td>
<td>0.42 1.05</td>
<td>0.72</td>
</tr>
<tr>
<td>VARMA(2,2)</td>
<td>complete</td>
<td>1.10 1.19</td>
<td>1.07 1.17</td>
<td>1.08</td>
</tr>
<tr>
<td></td>
<td>miss-5a</td>
<td>0.34 0.45</td>
<td>0.27 0.66</td>
<td>0.55</td>
</tr>
<tr>
<td></td>
<td>miss-5b</td>
<td>0.36 0.51</td>
<td>0.38 0.75</td>
<td>0.74</td>
</tr>
</tbody>
</table>

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 matri-
ces, 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, \( f: \mathbb{R}^{M \times N} \rightarrow \mathbb{R} \), then the \( N \times M \) matrix with \((i, j)\)-element \( \partial f / \partial \chi_{ij} \) will be denoted by \( f'(X) \) or \( df/dX \). If \( f \) is a vector valued function of a matrix, \( f: \mathbb{R}^{M \times N} \rightarrow \mathbb{R}^{m} \) then \( df/dX \) or \( f'(X) \) denotes the block matrix:

\[
\begin{bmatrix}
\frac{\partial f}{\partial \chi_{11}} & \ldots & \frac{\partial f}{\partial \chi_{1N}} \\
\vdots & \ddots & \vdots \\
\frac{\partial f}{\partial \chi_{M1}} & \ldots & \frac{\partial f}{\partial \chi_{MN}}
\end{bmatrix},
\]

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

\[
\begin{bmatrix}
\frac{\partial F}{\partial \chi_{11}} & \ldots & \frac{\partial F}{\partial \chi_{1N}} \\
\vdots & \ddots & \vdots \\
\frac{\partial F}{\partial \chi_{M1}} & \ldots & \frac{\partial F}{\partial \chi_{MN}}
\end{bmatrix}.
\]

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 \( \partial f_{ij}(X)/\partial \chi_{lc} \). 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 \( d(aF + \tilde{F})/dX = adF/dX + d\tilde{F}/dX \). We also have (where \( e_l \) is the \( l \)-th unit vector):

\[
[dX/dX]_{lc} = e_le_l^T, \tag{A.2}
\]

and, if \( G \) is another matrix function \( G: \mathbb{R}^{M \times N} \rightarrow \mathbb{R}^{m \times k} \), 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):

\[
\begin{align*}
[dX^T/dX]_{lc} &= e_le_l^T \\
[d(AX)/dX]_{lc} &= Ae_le_l^T & [d(XA)/dX]_{lc} &= e_le_l^TA \\
[d(AF)/dX]_{lc} &= AF'_{lc} & [d(FA)/dX]_{lc} &= F'_{lc}A \\
[d(XF)/dX]_{lc} &= XF'_{lc} + e_le_l^TF & [d(FX)/dX]_{lc} &= F'_{lc}X + Fe_le_l^T
\end{align*}
\]
where, in each case, $A$ is a constant matrix with dimensions compatible with those of $F$ and $X$. When $A$ is actually a vector, $A = a$, we have:

$$d(Xa)/dX = \begin{bmatrix} e_1 \\ \vdots \\ e_m \end{bmatrix} a^T. \quad (A.5)$$

and similarly, $d(a^TX)/dX = a[e_1^T \cdots e_n^T]$. If $n = 1$ and $F$ is vector-valued, $F = f = [f_1, \ldots, f_m]^T$, the $l$-th block-row of $d(Xf)/dX$ is $X[\partial f/\partial x_{1l} \cdots \partial f/\partial x_{nl}] + e_l f^T$ and the $c$-th block-column of $d(f^TX)/dX$ is $[\partial f/\partial x_{cl} \cdots \partial f/\partial x_{ml}]^T X + fe_c^T$. Furthermore:

$$[d(f^Tf)/dX]_{lc} = 2f_l^T(\partial f/\partial x_{lc}) \quad (A.6)$$

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:

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

(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),

$$dX^2/dx_{21} = X e_1 e_1^T + e_2 e_2^T X = \begin{bmatrix} x_{12} & 0 \\ x_{22} + x_{11} & x_{12} \end{bmatrix} \quad \text{and} \quad dX^2/dx_{12} = X e_2 e_2^T + e_1 e_1^T X = \begin{bmatrix} x_{21} & x_{11} + x_{22} \\ 0 & x_{21} \end{bmatrix}.$$

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

$$A(\partial y/\partial x_{lc}) = \partial b/\partial x_{lc} - A'_y y. \quad (A.8)$$

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

$$AF'_c = B'_c - A'_c F. \quad (A.9)$$
A.4 Derivative of Cholesky factorization

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

$$
S'_k = \begin{bmatrix}
S^r_t & s^r_{kk} \\
S^r_2 & t & S_3
\end{bmatrix},
L = \begin{bmatrix}
L_1 \\
L_2 \\
L_3 & v & L_2
\end{bmatrix}
$$

and $L'_k = \begin{bmatrix}
L'_1 \\
L'_2 \\
L'_3 & v' & L'_2
\end{bmatrix}$

then $2(u^Tu' + l_{kk}) = s^r_{kk}$ and $L'u' + L'u = s'$ so that

$$
L'u' = s' - L'\hat{u}
$$

(A.10)

and

$$
l'_k = (s^r_{kk}/2 - u^Tu')/l_{kk}. 
$$

(A.11)

These relations may be used iteratively for $k = 1, 2, \ldots$ to calculate $L'_k$ line by line, with $u'$ 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

$$
S_0 - (A_1^T S_1^T + \ldots + A_{p-1}^T S_{p-1}^T) - (A_p S_0 A_p^T + A_p S_1 A_{p-1}^T + \ldots + A_p S_{p-1} A_1^T) = G_0 + A_p G_p^T
given
$$

(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 vec$A$ for the columns of the lower triangle of $A$ placed one after another. A useful property here is vec$(ASB^T) = (B \otimes A) \text{vec}\ S$. Let $s_0 = \text{vec} S_0$, $g = \text{vec} G_0$ and denote the $k$-th column of $A_i$ with $a_{ik}$ and the $k$-th unit vector with $e_k$. Because $S_0$ is symmetric, taking the transpose of (B.1) gives with this notation:

$$
(I - A_p \otimes A_p) s_0 - (A_1 \otimes I + A_p \otimes A_{p-1}) s_1 - \ldots - (A_{p-1} \otimes I + A_p \otimes A_1) s_p = \text{vec} G_0^T + (A_p \otimes I) g_p
$$

(B.2)

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

$$
s_1 - \hat{A}_p s_0 - \hat{A}_p s_1 - \ldots - \hat{A}_p s_{p-1} = g_1
$$

$$
s_2 - \hat{A}_p s_1 - \hat{A}_p s_2 - \hat{A}_p s_{p-2} = g_2
$$

$$
\vdots
$$

$$
s_{p-1} - \hat{A}_p s_{p-2} - \ldots - \hat{A}_p s_0 - \hat{A}_p s_1 = g_{p-1}
$$

(B.3)
where $\tilde{A} = I \otimes A$ and $\tilde{A}_i$ is also an $r^2 \times r^2$ sparse block-matrix (which cannot be represented using $\otimes$):

$$\tilde{A}_i = \begin{bmatrix} a_{i1}e_1^T & \cdots & a_{i1}e_r^T \\ \vdots & \ddots & \vdots \\ a_{ir}e_1^T & \cdots & a_{ir}e_r^T \end{bmatrix}.$$ 

Together (B.2) and (B.3) provide $pr^2$ linear equations in the elements of $s_0, \ldots, s_{p-1}$, but one can (and should) take into account that $S_0$ is symmetric and $s_0$ contains therefore duplicated elements. Let $\tilde{S}_0$ be a lower triangular matrix such that $S_0 = \tilde{S}_0 + \tilde{S}_0^T$ (the diagonal elements of $\tilde{S}_0$ are halved compared with $S_0$) and let $\tilde{s}_0 = \text{vech} S_0$ (the $(r (r + 1)/2)$-vector obtained by removing the duplicated elements from $s_0$). Let also $J$ be an $r^2 \times r(r + 1)/2$ matrix such that post-multiplication with it removes columns $r + 1, 2r + 1, 2r + 2, 3r + 1, 3r + 2, 3r + 3, \ldots, r^2 - 1$. Then a term of the type $\tilde{A}_i s_i$ in (B.3) may be rewritten:

$$\tilde{A}_i s_i = (I \otimes A) s_i = \text{vec}(A A_i) = \text{vec}((\tilde{A}_i \tilde{A}_i^T + A_i A_i^T)) = (\tilde{A}_i + A_i) J \tilde{s}_i.$$  

(B.4)

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

$$(A_p \otimes A_p) s_i = (A_p \otimes A_p + \tilde{A}_p \tilde{A}_p) \text{vec}(S_0) = (A_p \otimes A_p + A_p \tilde{A}_p) J \tilde{s}_i,$$

and furthermore that $s_i = D \text{vec}(\tilde{S}_0)$ where $D$ is diagonal with $d_{ii} = 2$ when $i$ corresponds to a diagonal element in $S_0$ (i.e. $i = 1, r + 2, 2r + 3, \ldots$); otherwise $d_{ii} = 1$. The matrix $\tilde{A}_p \tilde{A}_p$ is a block-matrix with $(i, j)$-block equal to $a_{ij} a_{ij}^T$.

Finally, the upper triangle of (B.2) should be removed. These modifications result in $r^2 p - r(r - 1)/2$ equations in the same number of unknowns, the elements of $\tilde{s}_i, s_1, \ldots, s_p$; (B.2) becomes

$$J^T \left((D - A_p \otimes A_p - \tilde{A}_p \tilde{A}_p) J \tilde{s}_0 - \sum_{i=1}^{p-1} (A_i \otimes I + A_i \otimes A_{p-i}) s_i\right) = J^T \left(\text{vec} G_0^T + (A_p \otimes I) g_p\right)$$  

(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 $\varepsilon_t, \chi_t$ for $t=1, \ldots, h$ where $h = \max(p, q)$ one may draw $\varepsilon_t$ from $N(0,\Sigma)$ for $t = h+1, h+2, \ldots$ and apply (2.2) and (2.1) to obtain simulated values of $\chi_t$ 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 \( x' = (x_1^T, \ldots, x_h^T)^T \) have mean \( \mu' \) and covariance matrix \( S' \), and let \( C' = \text{cov}(x', \varepsilon') \). \( S' \), \( \Sigma' \) 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 \( \mu' \) is the \( rh \)-vector \( (\mu_1^T, \ldots, \mu_l^T)^T \). Starting values for \( x' \) may be drawn from \( N(\mu', \Sigma') \), and starting values for \( \varepsilon' \) (that are needed if there are moving average terms) may be drawn from the conditional distribution of \( \varepsilon' | x \), which is normal with expectation \( -\frac{1}{C S^{-1}} \) and covariance matrix \( \Sigma' - C'^T S^{-1} C' \). This conditional distribution may also be used to draw \( \varepsilon' \) when \( x_1, \ldots, x_h \) are given and \( \varepsilon_1, \ldots, \varepsilon_h \) are unknown, for example when forecasting with a moving average model. This procedure has been implemented in [Jonasson 200x].

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 \( \det(M + UV^T) = \det(M) \det(I + V^TM^{-1}U) \), 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

\[
\det(I_m + AB) = \det \left( C^{-1}C + C^{-1} \begin{pmatrix} I_k & 0 \\ 0 & 0 \end{pmatrix} D^{-1}D \begin{pmatrix} B_1 & B_2 \\ B_3 & B_4 \end{pmatrix} C \right) = \det(C^{-1}) \det \left( I_m + \begin{pmatrix} I_k & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} B_1 & B_2 \\ B_3 & B_4 \end{pmatrix} \right) \det C
\]

\[= \det \left( I_m + \begin{pmatrix} I_k + B_1 & B_2 \\ 0 & I_{m-k} \end{pmatrix} \right) = \det(I_k + B_1) = \det \left( \begin{pmatrix} I_k & 0 \\ B_3 & I_{n-k} \end{pmatrix} \right) \]

\[= \det \left( I_m + D \begin{pmatrix} B_1 & B_2 \\ B_3 & B_4 \end{pmatrix} CC^{-1} \begin{pmatrix} I_k & 0 \\ 0 & 0 \end{pmatrix} D^{-1} \right) \]

\[= \det(I_n + BA).\]

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


