Algorithm 878: Exact VARMA Likelihood and Its Gradient for Complete and Incomplete Data with Matlab

KRISTJAN JONASSON
University of Iceland

Matlab functions for the evaluation of the exact log-likelihood of VAR and VARMA time series models are presented (vector autoregressive moving average). The functions accept incomplete data, and calculate analytical gradients, which may be used in parameter estimation with numerical likelihood maximization. Allowance is made for possible savings when estimating seasonal, structured or distributed lag models. Also provided is a function for creating simulated VARMA time series that have an accurate distribution from term one (they are *spin-up* free). The functions are accompanied by a a simple example driver, a program demonstrating their use for real parameter fitting, as well as a test suite for verifying their correctness and aid further development. The article concludes with description of numerical results obtained with the algorithm.

Categories and Subject Descriptors: G.3 [**Probability and Statistics**]—*Multivariate statistics; Statistical software; Stochastic processes; Time series analysis*; G.4 [**Mathematical Software**]—*Algorithm design and analysis; Documentation; Efficiency; Matlab*; I.6 [**Simulation and Modelling**]: Miscellaneous; J.2 [**Computer Applications**]: Physical Sciences and Engineering; J.4 [**Computer Applications**]: Social and Behavioural Sciences—*Economics*

General Terms: Documentation, Verification, Algorithms

Additional Key Words and Phrases: Exact likelihood function, missing values, incomplete data, ARMA, VARMA, vector autoregressive moving average model.

ACM Reference Format:

Jonasson, K. 2008. Algorithm 878: Exact VARMA likelihood and its gradient for complete and incomplete data with Matlab. ACM Trans. Math. Softw. 35, 1, Article 6 (July 2008), 11 pages. DOI = 10.1145/1377603.1377609 http://doi.acm.org/10.1145/1377603.1377609

1. INTRODUCTION

Algorithm 878 consists of Matlab functions to aid in the analysis of multivariate time series models. There are three functions for evaluating exact

Author's address: K. Jonasson, Department of Computer Science, Faculty of Engineering, University of Iceland, Hjardarhaga 4, 107 Reykjavik, Iceland; email: jonasson@hi.is.

log-likelihood, a function for simulating time series, a suite of test functions for verifying the correctness of the other functions and a program demonstrating actual parameter fitting. A simple example driver is also included. The three log-likelihood evaluation functions are `varma_llc` for VARMA (vector autoregressive moving average) models with complete data, `varma_llm` for VARMA models with missing values and `var_ll` for VAR models with or without missing values. All three functions can optionally calculate the gradient of the log-likelihood, estimates of missing values, and estimates of associated residual or shock series.

The simulation function is named `varma_sim`, and it may be used to generate a single VARMA or VAR series, or several series at a time sharing the same parameters. One of the novelties of `varma_sim` is that the initial values are drawn from the appropriate distribution, so that throwing away the first part of the series to avoid spin-up effects is not needed.

The algorithm is an implementation of the methods described in the companion article, Jonasson and Ferrando [2008], and the programs (including variable names) follow very closely the notation used there.

The test suite implements *unit tests* for all functions and subfunctions, as far as is practical for a numerical package. The purpose is to ascertain the correctness of the coded algorithms, not to provide users with examples of how to use the package, which is provided for by the demonstration programs. Unit-testing is one of the ideas of *extreme programming*: write tests for everything, preferably before writing the actual algorithms being implemented; see, for example, George and Williams [2004].

Previously published programs for VARMA and VAR likelihood evaluation are the Fortran programs of Shea [1989] and Mauricio [1997] and the Matlab programs of Schneider and Neumaier [2001]. Attention should also be drawn to E^4 by J. Terceiro and others. It is a collection of Matlab functions for state-space estimation of econometric models. E^4 is distributed under the GNU license and available on the web along with a user manual: www.ucm.es/info/icae/e4. The software and manual have not been published, but there are some related publications listed on this Web page, including Terceiro [1990].

2. FUNCTION VALUE AND GRADIENT OF LOG-LIKELIHOOD

There are three functions for likelihood evaluation supplied: *var_ll* implements the savings described in Section 3.3 of the companion article for both the complete data and missing value cases, *varma_llc* implements the method of Section 2.2 and *varma_llm* the method of Section 3.1 in the companion article. The functions can also find gradients, residual estimates and missing value estimates, cf. Sections 3.2 and 4 of the companion article. When observations are missing, the functions accept the mean-vector as a proper model parameter (cf. the introduction to the companion paper), but the complete data calls assume a zero-mean model. To fit a non-zero-mean time-series, the mean-vector of the observations may be subtracted from each x_t at the outset. Help for all three functions is obtained by giving the command `help` *function* at the Matlab prompt (e.g. `help var_ll`).

2.1 VAR Models

A zero-mean VAR model describing a time series of values $\mathbf{x}_t \in \mathbb{R}^r$, $t = 1, \ldots, n$, is given by:

$$\mathbf{x}_t = \sum_{j=1}^{p} A_j \mathbf{x}_{t-j} + \boldsymbol{\varepsilon}_t \tag{1}$$

where the A_j's are $r \times r$ parameter matrices and the $\boldsymbol{\varepsilon}_t$'s are r-variate $N(\mathbf{0}, \Sigma)$ uncorrelated in time. To evaluate the exact log-likelihood function associated with (1) when no observations are missing use the Matlab call:

$$[\texttt{ll}, \texttt{ok}] = \texttt{var\_ll}(\texttt{X}, \texttt{A}, \texttt{Sig})$$

where `A` is an $r \times rp$ matrix containing $[A_1 \ \ldots \ A_p]$, `Sig` is $r \times r$ and symmetric containing Σ, and `X` is an $r \times n$ matrix with $\mathbf{x}_t$ in its t-th column. Let $\theta \in \mathbb{R}^{n_\theta}$ with $n_\theta = pr^2 + r(r+1)/2$ denote the vector of parameters, i.e. the elements of $A_1, \ldots, A_p$ (column by column) and the lower triangle of Σ. If they describe a stationary model, `ll` will return a scalar with the value of the log-likelihood function $l(\theta)$ and the logical variable `ok` will return true, but if the model is nonstationary `lld` will be 0 and `ok` will be false. To calculate the $1 \times n_\theta$ gradient $l'(\theta)$ in `lld` or the (maximum likelihood) estimate of the residuals (or shocks) $\boldsymbol{\varepsilon}_t$ in `res` use the calls:

$$[\texttt{ll}, \texttt{ok}, \texttt{lld}] = \texttt{var\_ll}(\texttt{X}, \texttt{A}, \texttt{Sig})$$
$$[\texttt{ll}, \texttt{ok}, \texttt{res}] = \texttt{var\_ll}(\texttt{X}, \texttt{A}, \texttt{Sig}, \text{`}\texttt{res}\text{'}).$$

A non-zero-mean VAR model may be written:

$$\mathbf{x}_t - \boldsymbol{\mu} = \sum_{j=1}^{p} A_j (\mathbf{x}_{t-j} - \boldsymbol{\mu}) + \boldsymbol{\varepsilon}_t \tag{2}$$

If `X`, `A` and `Sig` are as before, `mu` contains $\boldsymbol{\mu}$ and `miss` is an $r \times n$ logical matrix, which is true in locations of missing values, the Matlab call:

$$[\texttt{ll}, \texttt{ok}] = \texttt{var\_ll}(\texttt{X}, \texttt{A}, \texttt{Sig}, \texttt{mu}, \texttt{miss})$$

will return the log-likelihood value in `ll` (or zero `ll` and false `ok` if the model is non-stationary). To calculate the gradient, residuals, or residuals and maximum likelihood estimates of missing values use the calls

$$[\texttt{ll}, \texttt{ok}, \texttt{lld}] = \texttt{var\_ll}(\texttt{X}, \texttt{A}, \texttt{Sig}, \texttt{mu}, \texttt{miss})$$
$$[\texttt{ll}, \texttt{ok}, \texttt{res}] = \texttt{var\_ll}(\texttt{X}, \texttt{A}, \texttt{Sig}, \texttt{mu}, \texttt{miss}, \text{`}\texttt{res}\text{'})$$
$$[\texttt{ll}, \texttt{ok}, \texttt{res}, \texttt{xm}] = \texttt{var\_ll}(\texttt{X}, \texttt{A}, \texttt{Sig}, \texttt{mu}, \texttt{miss}, \text{`}\texttt{res\_miss}\text{'}).$$

($\boldsymbol{\mu}$ is placed at the end of θ and n_θ is now $pr^2 + r(r+3)/2$).

2.2 Complete Data VARMA Models

A zero-mean VARMA model for $\mathbf{x}_t \in \mathbb{R}^r$, $t = 1, \ldots, n$, is given by

$$\mathbf{x}_t = \sum_{j=1}^{p} A_j \mathbf{x}_{t-j} + \mathbf{y}_t \tag{3}$$

where $\mathbf{y}_t = \boldsymbol{\varepsilon}_t + \sum_{j=1}^{q} B_j \boldsymbol{\varepsilon}_{t-j}$, 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. If X, A and Sig are as in Section 2.1 and B contains $[B_1 \ \ldots \ B_q]$, the Matlab call

$$[\texttt{ll}, \texttt{ok}] = \texttt{varma\_llc}(\texttt{X}, \texttt{A}, \texttt{B}, \texttt{Sig})$$

will return the likelihood function value in ll and ok will be true unless the model is non-stationary, then ok will be false. The calls

$$[\texttt{ll}, \texttt{ok}, \texttt{lld}] = \texttt{varma\_llc}(\texttt{X}, \texttt{A}, \texttt{Sig})$$
$$[\texttt{ll}, \texttt{ok}, \texttt{res}] = \texttt{varma\_llc}(\texttt{X}, \texttt{A}, \texttt{Sig}, \text{'}\texttt{res}\text{'})$$

return in addition the $1 \times (p+q)r^2 + r(r+1)/2$ gradient $l'(\theta)$ in lld or the residual estimates in res, where θ consists of the parameters: the columns of the A_j's followed by the columns of the B_j's followed by the columns of the lower triangle of Σ.

2.3 Missing Value VARMA Models

Let $\boldsymbol{\mu}$ be the expected value of $\mathbf{x}_t \in \mathbb{R}^r$ and let other model parameters as well as $\mathbf{y}_t$ and $\boldsymbol{\varepsilon}_t$ be as in Section 2.2. A nonzero-mean VARMA model for $\mathbf{x}_t$, $t = 1, \ldots, n$, is given by

$$\mathbf{x}_t - \boldsymbol{\mu} = \sum_{j=1}^{p} A_j(\mathbf{x}_{t-j} - \boldsymbol{\mu}) + \mathbf{y}_t \tag{4}$$

If X, A, B, and Sig are as in Section 2.2, mu contains $\boldsymbol{\mu}$ and miss is an $r \times n$ logical matrix that is true in locations of missing values, then the Matlab call

$$[\texttt{ll}, \texttt{ok}] = \texttt{varma\_llm}(\texttt{X}, \texttt{A}, \texttt{B}, \texttt{Sig}, \texttt{mu}, \texttt{miss})$$

will return the likelihood function value in ll and ok will be true unless the model is nonstationary. The calls

$$[\texttt{ll}, \texttt{ok}, \texttt{lld}] = \texttt{varma\_llm}(\texttt{X}, \texttt{A}, \texttt{B}, \texttt{Sig}, \texttt{mu}, \texttt{miss})$$
$$[\texttt{ll}, \texttt{ok}, \texttt{res}] = \texttt{varma\_llm}(\texttt{X}, \texttt{A}, \texttt{B}, \texttt{Sig}, \texttt{mu}, \texttt{miss}, \text{'}\texttt{res}\text{'})$$
$$[\texttt{ll}, \texttt{ok}, \texttt{res}, \texttt{xm}] = \texttt{varma\_llm}(\texttt{X}, \texttt{A}, \texttt{B}, \texttt{Sig}, \texttt{mu}, \texttt{miss}, \text{'}\texttt{res\_miss}\text{'})$$

return in addition the $1 \times (p+q)r^2 + r(r+3)/2$ gradient $l'(\theta)$ in lld, residual estimates in res, and the last one returns maximum likelihood estimates of missing values in xm, where $\boldsymbol{\mu}$ has been appended to the θ of Section 2.2.

2.4 Model Parameters Given by a Function

Let $\theta = g(\phi)$ where $\phi \in \mathbb{R}^{n_\phi}$ and let J_g be the $n_\theta \times n_\phi$ Jacobian of g as in Section 4.3 of the companion paper. To realize the savings discussed there, use one of the calls

$$[\texttt{ll}, \texttt{ok}, \texttt{lld}] = \texttt{var\_ll}(\texttt{X}, \texttt{A}, \texttt{Sig}, \texttt{J})$$
$$[\texttt{ll}, \texttt{ok}, \texttt{lld}] = \texttt{var\_ll}(\texttt{X}, \texttt{A}, \texttt{Sig}, \texttt{mu}, \texttt{miss}, \texttt{J})$$
$$[\texttt{ll}, \texttt{ok}, \texttt{lld}] = \texttt{varma\_llc}(\texttt{X}, \texttt{A}, \texttt{Sig}, \texttt{J})$$
$$[\texttt{ll}, \texttt{ok}, \texttt{lld}] = \texttt{varma\_llm}(\texttt{X}, \texttt{A}, \texttt{B}, \texttt{Sig}, \texttt{mu}, \texttt{miss}, \texttt{J})$$

where J contains J_g. A partial variable change is also possible; c.f. the help text of the functions. To take an example, assume a distributed lags model,

$$\mathbf{x}_t = C\sum_{j=1}^{3} b_j \mathbf{x}_{t-j} + \boldsymbol{\varepsilon}_t$$

where the b_j's are fixed constants and the model parameters consist of the $r \times r$ matrix C together with the shock covariance matrix Σ. To evaluate the likelihood and its gradient efficiently the following Matlab code may be used:

```
A = [b(1) * C b(2) * C b(3) * C];
I = eye(3 * r);
JC = [b (1) * I; b (2) * I; b (3) * I];
JSig = eye (r * (r + 1)/2);
J = blkdiag (JC, JSig);
[ll, ok, lld] = var_ll(X, A, Sig, J);
```

3. SIMULATION

Simulation of VARMA time series has many applications, for example, to create test data for modelling methods, analyze such methods, and forecast with fitted models.

3.1 Spin-Up Free Simulation

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 the formulae (2) and (1) in the companion article 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 $C' = \mathrm{cov}(\mathbf{x}', \boldsymbol{\varepsilon}')$. S', Σ' and C' are given with (7) and (8) in Article 5 and solution of the vector-Yule-Walker equations (10) applying (18) 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 $\Sigma' - C'^{T} S^{-1} C'$. 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.

3.2 A Simulation Function

The Matlab function `varma_sim` will generate a random VARMA time series for a specified model. If `A`, `B` and `Sig` are as above, then the calls

$$\texttt{X} = \texttt{varma\_sim(A, [], Sig, n)}$$
$$\texttt{X} = \texttt{varma\_sim(A, B, Sig, n)}$$

generate a single zero-mean n-term series modelled by (1) or (3) in the $r \times n$ matrix `X`. The calls

$$\texttt{X} = \texttt{varma\_sim(A, [], Sig, n, [], M)}$$
$$\texttt{X} = \texttt{varma\_sim(A, B, Sig, n, [], M)}$$

will create `M` such series. When $r = 1$ `X` will be $n \times M$ and when $r > 1$ it will be $r \times n \times M$. To generate non-zero-mean series as modelled by (2) or (4) use the calls

$$\texttt{X} = \texttt{varma\_sim(A, B, Sig, n, mu)}$$
$$\texttt{X} = \texttt{varma\_sim(A, B, Sig, n, mu, M)},$$

possibly with empty `B`. The series are started using the procedure described in Section 3.1 and when moving average terms are present, the initial shocks are also drawn as described there.

It is also possible to specify terms to start the series using

$$\texttt{X} = \texttt{varma\_sim(A, B, Sig, n, mu, M, X0)}$$

where `X0` has r rows and at least $\max(p, q)$ columns. All the generated series will begin with the last $\max(p, q)$ columns of `X0` and the corresponding shocks are drawn as explained above. As before, `A`, `B` and/or `mu` may be empty.

The shocks used for the generation may be obtained by specifying a second return parameter: `[X,eps] = varma_sim(...)`. The dimension of `eps` will be same as that of `X`.

4. DEMONSTRATION

4.1 Demonstration of Likelihood Calculation

A simple example driver, `example_driver.m`, illustrates the use of the three log-likelihood evaluating functions as well as simulation. The driver calculates the log-likelihood of two models, a VAR(1) model and a VARMA(1,1) model, both of them with $r = 2$ and $n = 12$. It also produces two simulated series of length 5.

4.2 Demonstration of Parameter Estimation

A suite of programs demonstrating the use of the package for actual model fitting has been gathered in one file, `demorun.m`. There are two subfunctions for two types of demonstration:

a) VAR(p) and VARMA(p, q) modelling with simulated data (obtained with `varma_sim`), both with and without missing values. These are carried out by the subfunction `demov`.

b) Modeling of real data using two constrained models is done by the subfunction `demod`. The data are annual mean temperatures at 3 Icelandic meteorological stations 1799–2006, cf. [Hanna et al. 2004]. The two models are a combined lower triangular and diagonal VAR-model:

$$\mathbf{x}_t - \boldsymbol{\mu} = L(\mathbf{x}_{t-1} - \boldsymbol{\mu}) + D_1(\mathbf{x}_{t-2} - \boldsymbol{\mu}) + D_2(\mathbf{x}_{t-3} - \boldsymbol{\mu}) + \boldsymbol{\varepsilon}_t \tag{5}$$

where L is lower triangular and D_1 and D_2 are diagonal, and a distributed lags VAR-model:

$$\mathbf{x}_t - \boldsymbol{\mu} = A(\mathbf{x}_{t-1} - \boldsymbol{\mu}) + 0.5A(\mathbf{x}_{t-2} - \boldsymbol{\mu}) + \boldsymbol{\varepsilon}_t \tag{6}$$

where A is a general matrix. In both cases the $\boldsymbol{\varepsilon}_t$'s are 3-variate $N(\mathbf{0}, \Sigma)$.

The parameters are estimated by maximizing the log-likelihood function using the BFGS-method. There are two choices for an optimization routine: `fminunc` from Matlab's optimization toolbox, and the function `ucminf` described in Nielsen [2000] and available freely in http://imm.dtu.dk/~hbn/immoptibox. Before running the demonstrations, one of these must be installed.

Issuing one of the commands

```
demorun('fminunc')
demorun('ucminf')
```

fits six models, four of type a) and the two models (5) and (6). To make the demonstration run quickly, small models have been chosen. For a) these are a VAR(2) model with $r = 3$, $n = 400$ and complete data, a VAR(2) model with $r = 3$, $n = 200$ and 5% of the values missing, and two VARMA(1, 1) models with $r = 2$, $n = 200$, one with complete data and the other with 5% missing. For (5) and (6) the data before 1860 is omitted, also to enable a quick run. The results are models with $p = 2$ and 3, $r = 3$, $n = 146$ and 6.6% of the values missing. All these sizes are easily changed by editing the function. A data file with the temperature series, as well as pdf files with the output of `demorun('ucminf')` and the source code of `demorun.m` accompany the program package.

5. TESTING

The programs in the test suite are of two types: primary tests, for testing the four main functions discussed above, and secondary tests, that test individual components (subfunctions, helper functions) of the main functions. To verify the correctness of the main functions it is only necessary to examine and run the primary tests. The secondary tests were written as an aid in developing the program suite. They are included for completeness, and as an aid for possible future development and changes. The primary tests are:

```
test_varma_llc
test_varma_llc_deriv
test_var_ll
test_var_ll_jac
test_varma_llm
test_varma_jac
test_varma_llm_deriv
test_varma_sim
```

The correctness of `varma_llc` is checked against direct likelihood evaluation with Equation (3) of the companion article. The function `varma_llm` is checked against `varma_llc` for complete data, and against direct evaluation with Equation (4) of the companion article for missing values, and `var_ll` is simply compared with `varma_llm`. Gradient calculation and the Jacobian feature (see Section 2.4) are checked by comparing with numerical differentiation. All tests are carried out for several different test cases with a range of values of p, q and r. Finally, the testing of `varma_sim` is accomplished by comparing data expectations and covariances of generated series with theoretical ones. All the primary tests may be run via the Matlab script `TEST_PRIMARY`, and with `TEST_ALL` the secondary tests are also run.

A comparison of `varma_llc` with calcuations from Algorithm AS311 of Mauricio [1997] was also carried out and an agreement to about 15 decimal digits was observed. The programs used for this comparison are included, together with their output.

6. NUMERICAL EXPERIMENTS

6.1 Likelihood Maximization

The primary use of likelihood evaluation is to estimate model parameters by maximizing the likelihood function. The program described in Section 4.2 demonstrates such parameter estimation. Choosing the *ucminf* optimization routine, the parameter estimates for the four models of the `demov` subfunction 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 took about 60–70% longer. The estimation of the two meteorological models of the `demod` subfunction took 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.

6.2 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 6.1). Table I shows the runtime 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 in the companion article are realized, and for complete data the solution to the vector-Yule-Walker

Table I. Runtime 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

equations and the calculation of $\mathbf{w}$ and $\mathbf{z}$ 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.

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 $\hat{\mathrm{V}}$ and $\hat{\Lambda}_{\mathrm{om}}$, 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 (18) and (19) in Section 3.1 in the companion article. This ranges from a factor of 1.26 to a factor of 2.61, the average being 1.89.

6.3 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

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

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).

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 the companion paper 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.

REFERENCES

George, B. and Williams, L. 2004. A structured experiment of test-driven development. *Inform. Softw. Tech. 46*, 5, 337–342.

Hanna, E., Jonsson, T., and Box, J. E. 2004. An analysis of Icelandic climate since the nineteenth century. *Int. J. Climatology 24*, 10, 1193–1210.

Jonasson, K. and Ferrando, S. E. 2008. Evaluating exact VARMA likelihood and its gradient when data are incomplete. *ACM Trans. Math Softw. 35*, 1.

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

Nielsen, H. B. 2000. UCMINF—An algorithm for unconstrained, nonlinear optimization. Tech, Rep. IMM-REP-2000-19, Department of Mathematical Modelling, Technical University of Denmark.

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.

TERCEIRO, J. 1990. *Estimation of Dynamic Econometric Models with Errors in Variables*. Springer-Verlag, Berlin, Germany.

Received August 2006; revised March 2007, September 2007; accepted October 2007