Algorithm xxx: 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.

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.

1. INTRODUCTION

Algorithm xxx consists of Matlab functions to aid in the analysis of multivariate time series models. There are three functions for evaluating exact 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 paper [Jonasson and Ferrando 200 xxx] and the programs (including variable names) follow very closely the notation used there. The companion paper also describes numerical experiments carried out with the Matlab functions.

The test suite implements *unit tests* for all functions and sub-functions, 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

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

[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 paper 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 paper. The functions can also find gradients, residual estimates and missing value estimates, c.f. sections 3.2 and 4 of the companion paper. When observations are missing, the functions accept the mean-vector as a proper model parameter (c.f. 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:

```
[ll,ok] = var_ll(X,A,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 non-stationary `ll` will be 0 and `ok` will be false. To calculate the $1 \times n_\theta$ gradient $l'(\theta)$ in `ll` or the (maximum likelihood) estimate of the residuals (or shocks) $\boldsymbol{\varepsilon}_t$ in `res` use the calls:

```
[ll,ok,lld] = var_ll(X,A,Sig)
[ll,ok,res] = var_ll(X,A,Sig,'res').
```

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:

```
[ll,ok] = var_ll(X,A,Sig,mu,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

```
[ll,ok,lld] = var_ll(X,A,Sig,mu,miss)
[ll,ok,res] = var_ll(X,A,Sig,mu,miss,'res')
[ll,ok,res,xm] = var_ll(X,A,Sig,mu,miss,'res_miss').
```

($\boldsymbol{\mu}$ 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

```
[ll,ok] = varma_llc(X,A,B,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

```
[ll,ok,lld] = varma_llc(X,A,Sig)
[ll,ok,res] = varma_llc(X,A,Sig,'res')
```

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 non-zero-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 which is true in locations of missing values then the Matlab call

```
[ll,ok] = varma_llm(X,A,B,Sig,mu,miss)
```

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

```
[ll,ok,lld] = varma_llm(X,A,B,Sig,mu,miss)
[ll,ok,res] = varma_llm(X,A,B,Sig,mu,miss,'res')
[ll,ok,res,xm] = varma_llm(X,A,B,Sig,mu,miss,'res_miss')
```

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

```
[ll,ok,lld] = var_ll(X,A,Sig,J)
[ll,ok,lld] = var_ll(X,A,Sig,mu,miss,J)
[ll,ok,lld] = varma_llc(X,A,Sig,J)
[ll,ok,lld] = varma_llm(X,A,B,Sig,mu,miss,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

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

```
X = varma_sim(A,[],Sig,n)
X = 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

```
X = varma_sim(A,[],Sig,n,[],M)
X = 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

```
X = varma_sim(A,B,Sig,n,mu)
X = varma_sim(A,B,Sig,n,mu,M),
```

possibly with empty `B`. The series are started using the procedure described in the second paragraph of Appendix D in the companion paper, 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

```
X = 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) Modelling 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, Jonsson and Box 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/imm-optibox. 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. 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 (2.3) of the companion paper. The function `varma_llm` is checked against `varma_llc` for complete data, and against direct evaluation with equation (2.4) of the companion paper 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.

REFERENCES

GEORGE, B. AND WILLIAMS, L. 2004. A structured experiment of test-driven development. *Information And Software Technology 46*, 5, 337–342.

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.

JONASSON, K. AND FERRANDO, S. E. 200 xxx, Evaluating exact VARMA likelihood and its gradient when data are incomplete. *ACM Trans. Math Softw. xxx*, xxx, xxx–xxx.

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. Report 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.