Some Computational Aspects of Exact Maximum Likelihood Estimation of Time Series Models

José Alberto Mauricio

Departamento de Fundamentos del Análisis Económico II, Facultad de Ciencias Económicas y Empresariales, Universidad Complutense de Madrid, Campus de Somosaguas, 28223-Madrid, Spain. (E-mail: eccua03@sis.ucm.es.)

Keywords. Computer programs, exact maximum likelihood, time series models

1 Introduction

It is well known (Ansley and Newbold 1980; Hillmer and Tiao 1979) that exact maximum likelihood estimation (EMLE) of time series models is usually preferable to other approximate estimation criteria. This is especially true in the case of small- to moderate-sized samples and/or parameters close to the boundaries of the admissible regions. Instead of pursuing this issue further, this paper focuses on some relevant computational aspects concerning the numerical maximization of the exact likelihood function of several time series models. The range of models considered here covers, among other more usual specifications, a new kind of seasonal univariate autoregressive-moving average (ARMA) models, single- and multiple-output transfer-function-noise models, vector ARMA models and, in general, time series models with parameters subject to certain constraints.

A unified framework for EMLE of these models is presented throughout the next sections. The basic idea is that of casting the model to be estimated into a standard vector ARMA(p,q) specification, (i) whose parameters are linear or nonlinear functions of the parameters appearing in the model to be estimated, and (ii) to which the most recent and efficient estimation methods (Shea 1984, 1989; Mauricio 1995, 1996) can be applied. The main advantage of working within the vector ARMA framework lies in the fact that a single algorithm can be used to estimate many apparently different models. Thus, the development of some guidelines on casting time series models into a standard vector ARMA(p,q) specification, in conjunction with an operational design of EMLE algorithms for vector ARMA models, are key steps in writing new time series analysis software that allows for working in practice with more than a few traditional models.

In order to clarify and give these ideas a practical sense, a few suggestions are provided in Section 2 on how to write non-standard time series models as standard vector ARMA(p,q) models. Then, in Section 3 some guidelines are given on writing computer programs that use efficiently the ideas mentioned above; in particular, it is shown that a straightforward modular design can be implemented in order to write expandable and easy-to-use code for estimating an almost unlimited range of time series models. In Section 4 some computationally relevant problems that arise during EMLE of vector ARMA models are considered in brief, including the most appropriate methods for detecting and dealing with situations of non-stationarity and/or non-invertibility. Recent applications of these ideas and some guidelines for future research are summarized in Section 5.

2 The Multivariate ARMA Framework

Consider a sample of size N on an M-dimensional time series $\mathbf{z}_t$ ($t = 1, \ldots, N$). Almost every linear statistical model for $\mathbf{z}_t$ can be expressed as a standard vector ARMA(p,q) model

$$\mathbf{\Phi}(B)\tilde{\mathbf{w}}_t = \mathbf{\Theta}(B)\mathbf{a}_t \ , \tag{2.1}$$

where $\tilde{\mathbf{w}}_t = \mathbf{w}_t - \boldsymbol{\mu}$ ($t = 1, \ldots, n$) and $\mathbf{w}_t$ is an $m\times 1$ vector; $\mathbf{\Phi}(B) = \mathbf{I} - \mathbf{\Phi}_1 B - \cdots - \mathbf{\Phi}_p B^p$, $\mathbf{\Theta}(B) = \mathbf{I} - \mathbf{\Theta}_1 B - \cdots - \mathbf{\Theta}_q B^q$; B is the back shift operator; $\mathbf{\Phi}_i$ ($i = 1, \ldots, p$), $\mathbf{\Theta}_i$ ($i = 1, \ldots, q$) and $\boldsymbol{\mu}$ are $m\times m$, $m\times m$ and $m\times 1$ parameter matrices, respectively; the $\mathbf{a}_t$'s are $m\times 1$ random vectors identically and independently distributed as $N(\mathbf{0}, \sigma^2\mathbf{Q})$, with $\sigma^2 > 0$ and $\mathbf{Q}$ ($m\times m$) symmetric and positive definite; and, finally, n and m are related to N and M in a known way.

One may think of every component of $\mathbf{w}_t$ ($t = 1, \ldots, n$), $\mathbf{\Phi}_i$ ($i = 1, \ldots, p$), $\mathbf{\Theta}_i$ ($i = 1, \ldots, q$), $\boldsymbol{\mu}$ and $\mathbf{Q}$ as being an explicit function of a $k\times 1$ vector $\mathbf{x}$, whose elements either are the parameters of the time series model considered, or are related to them in a known way; in the case of $\mathbf{w}_t$, it can also be thought of as depending (primarily) on the data $\mathbf{z}_t$.

To illustrate, consider a sample of 100 observations on $\mathbf{z}_t = (z_{1t}, z_{2t})^T$, so that $N = 100$, $M = 2$, and suppose that an analyst specifies the following transfer-function-noise model for $\mathbf{z}_t$:

$$\begin{aligned} \ln z_{1t} &= \beta_1\xi_{1t} + \frac{\omega_0}{1-\delta_1 B}\ln z_{2t} + \frac{1}{(1-\phi_1 B)\nabla}\,u_{1t} \ ; \\ \nabla \ln z_{2t} &= (1-\theta_2 B)\,u_{2t} \ , \end{aligned} \tag{2.2}$$

where ξ_{1t} represents a deterministic variable, $\nabla = 1 - B$, and u_{1t} and u_{2t} are independent white noise disturbances with variances σ_1^2 and σ_2^2, respectively. After some algebraic manipulation, (2.2) can be written as follows:

$$\begin{aligned} &\begin{bmatrix} (1-\phi_1 B)(1-\delta_1 B) & \omega_0 - \omega_0(1-\phi_1 B) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \nabla(\ln z_{1t} - \beta_1\xi_{1t}) \\ \nabla \ln z_{2t} \end{bmatrix} = \\ &\qquad \begin{bmatrix} 1-\delta_1 B & \omega_0(1-\theta_2 B) - \omega_0(1-\delta_1 B) \\ 0 & 1-\theta_2 B \end{bmatrix} \begin{bmatrix} u_{1t} + \omega_0 u_{2t} \\ u_{2t} \end{bmatrix} . \end{aligned} \tag{2.3}$$

Thus, taking $n = N - 1 = 99$, $m = M = 2$, and letting $\mathbf{\Phi}(B)$, $\mathbf{\Theta}(B)$, $\mathbf{w}_t$ and $\mathbf{a}_t$ represent, respectively, the two 2×2 polynomial matrices and the two 2×1 vectors appearing in (2.3), it turns out that (2.3) has the same form as (2.1), with $\boldsymbol{\mu} = \mathbf{0}$ and:

$$E[\mathbf{a}_t \mathbf{a}_t^T] = \sigma^2 \mathbf{Q} = \sigma_2^2 \begin{bmatrix} (\sigma_1^2/\sigma_2^2) + \omega_0^2 & \omega_0 \\ \omega_0 & 1 \end{bmatrix} . \tag{2.4}$$

Hence, EMLE of β_1, ω_0, δ_1, ϕ_1, θ_2, σ_1^2 and σ_2^2, the parameters of the original specification (2.2), can be performed by maximizing a concentrated log likelihood for (2.1) (Shea 1984; Mauricio 1995) as a function of $\mathbf{x} = (\beta_1, \omega_0, \delta_1, \phi_1, \theta_2, \eta)^T$ only. (Note that $\eta = \sigma_1^2/\sigma_2^2$ and, therefore, $k = 6$.)

Although the algebra leading from (2.2) to (2.3)-(2.4) becomes more involved when considering more complex models, the basic idea of casting the specified model into a standard vector ARMA(p,q) model remains unchanged. Clearly, the casting algebra will vary from model to model, so that user involvement at this stage is unavoidable. However, it is also true that the casting process can be automated in some instances for production purposes; this is possible, for example, in the case of scalar and vector ARMA models, including both multiplicative Box-Jenkins and generalized seasonal models with frequency restrictions (Gallego 1995). Thus, computer programs allowing for high productivity when dealing with usual models, as well as providing enough flexibility for dealing with non-usual and/or complex models, are very valuable tools for EMLE of many time series models. These dual-purpose programs can be coded through a modular design of the kind described in the next section.

3 A Modular System of Estimation Algorithms

In order to perform EMLE of the parameter vector $\mathbf{x}$ that makes up the standard vector ARMA(p,q) model (2.1), one may minimize numerically, starting at an admissible initial guess $\mathbf{x}_0$, the following scaled objective function:

$$F(\mathbf{x}) = \frac{\Pi_1(\mathbf{x})}{\Pi_{10}} \, \frac{\Pi_2(\mathbf{x})}{\Pi_{20}} , \tag{3.1}$$

where:

$$\Pi_1(\mathbf{x}) = (\tilde{\mathbf{w}}^T \boldsymbol{\Sigma}^{-1} \tilde{\mathbf{w}})^m , \quad \Pi_2(\mathbf{x}) = |\boldsymbol{\Sigma}|^{1/n} , \tag{3.2}$$

$\Pi_{10} = \Pi_1(\mathbf{x}_0)$, $\Pi_{20} = \Pi_2(\mathbf{x}_0)$, $\tilde{\mathbf{w}} = (\tilde{\mathbf{w}}_1^T, ..., \tilde{\mathbf{w}}_n^T)^T$, and $\boldsymbol{\Sigma} = E[\tilde{\mathbf{w}}\tilde{\mathbf{w}}^T]$. (Note that both $\tilde{\mathbf{w}}$ and, mainly, $\boldsymbol{\Sigma}$ depend on the parameter vector $\mathbf{x}$.) On convergence, a sample estimate of the covariance matrix of the exact maximum likelihood estimator is given by $2F(\bar{\mathbf{x}})[n\boldsymbol{\nabla}^2 F(\bar{\mathbf{x}})]^{-1}$, where $\boldsymbol{\nabla}^2 F(\bar{\mathbf{x}})$ represents the hessian matrix of (3.1) evaluated at the final estimate $\bar{\mathbf{x}}$ (Mauricio 1995).

The computation of (3.2) at every iteration of the minimization algorithm, can be carried out through any of the currently available methods for evaluating the exact likelihood function of vector ARMA models; Shea (1989) and Mauricio

(1996) are reasonably good choices here. The numerical minimization procedure can be any of the many currently available methods, although a quasi-Newton method based on the factorized version of the BFGS formula is most advisable.

With these ideas in mind, the design of EMLE programs for time series models can be made up of five modules: (1) a user module (USER), (2) a driver module (DRIVER), (3) a module for the computation of the exact log-likelihood of vector ARMA models (ELFVARMA), (4) a numerical optimization module (OPT), and (5) a numerical linear algebra module (LALG). The following operations should be performed within each of them:

1 Module USER:
   1.1 Set k (scalar: number of parameters) and $\mathbf{x}_0$ ($k\times 1$ vector: initial guess).
   1.2 Implement routine CAST [cast time series model into model (2.1)].
   1.3 Call module DRIVER and process output on return.
2 Module DRIVER:
   2.1 Implement routine OBJFUNC [set scaled objective function (3.1)].
   2.2 Put $\mathbf{x}_0$ into standard vector ARMA structure (CAST).
   2.3 Compute (3.2) at $\mathbf{x}_0$ (ELFVARMA): initialize Π_{10} and Π_{20}.
   2.4 Minimize (3.1) starting at $\mathbf{x}_0$ (OPT): obtain final estimate $\bar{\mathbf{x}}$.
   2.5 Compute sample covariance matrix at $\bar{\mathbf{x}}$.
   2.6 Put $\bar{\mathbf{x}}$ into standard vector ARMA structure (CAST).
   2.7 Compute (3.2) and residuals at $\bar{\mathbf{x}}$ (ELFVARMA) and return.
3 Module ELFVARMA: Check for appropriateness of parameter values and compute (3.2).
4 Module OPT: Minimize numerically a k-variable real function.
5 Module LALG: Perform some linear algebra computations for modules ELFVARMA and OPT.

The whole EMLE process is driven by module DRIVER. Implementation of routine OBJFUNC for computing (3.1) at every value of $\mathbf{x}$ is accomplished in two steps: (i) put $\mathbf{x}$ into standard vector ARMA structure (CAST), and (ii) compute (3.2) at $\mathbf{x}$ (ELFVARMA) and obtain (3.1). (Note that OBJFUNC contains the k-variable real function to be minimized through module OPT.) The flow of module DRIVER is otherwise quite simple. With regard to module ELFVARMA, this is an essential one in that not only must it perform an efficient and accurate computation of (3.2), but also has to check for stationarity and invertibility of the resulting standard model (2.1) (see Section 4 for details). Matrix operations needed by modules ELFVARMA and OPT (Cholesky factorization, forward and backward substitution, solution of general linear equation systems and computation of eigenvalues) are performed within module LALG. At this point, it is interesting to note that once modules DRIVER, ELFVARMA, OPT and LALG have been coded, the user has to worry about nothing but module USER; note also that modules ELFVARMA and OPT may be implemented in different versions, all of which should be open to the analyst from module USER.

What makes the above design really modular and useful is the fact that the user may specify as many CAST routines as needed. To illustrate, consider the example in Section 2; EMLE of model (2.2) requires setting $k = 6$, $\mathbf{x}_0$ [an initial guess for $\mathbf{x} = (x_1, x_2, x_3, x_4, x_5, x_6)^T = (\beta_1, \omega_0, \delta_1, \phi_1, \theta_2, \eta)^T$] and, from (2.3)-(2.4), specifying a CAST routine implementing the following assignments:

$$m = M\ (= 2),\ n = N - 1\ (= 99),\ p = 2,\ q = 1,$$

$$\mathbf{\Phi}_1 = \begin{bmatrix} x_3 + x_4 & -x_2 x_4 \\ 0.0 & 0.0 \end{bmatrix},\quad \mathbf{\Phi}_2 = \begin{bmatrix} -x_3 x_4 & 0.0 \\ 0.0 & 0.0 \end{bmatrix},$$

$$\mathbf{\Theta}_1 = \begin{bmatrix} x_3 & -x_2 (x_3 - x_5) \\ 0.0 & x_5 \end{bmatrix},$$

$$\mathbf{Q} = \begin{bmatrix} x_6 + x_2^2 & x_2 \\ x_2 & 1.0 \end{bmatrix},\quad \mathbf{w}_t = \begin{bmatrix} \nabla (\ln z_{1t} - x_1 \xi_{1t}) \\ \nabla \ln z_{2t} \end{bmatrix} \ (t = 1, ..., n)\ .$$

On return from module DRIVER, exact maximum likelihood estimates for β_1, ω_0, δ_1, ϕ_1, θ_2 and $\eta\ (= \sigma_1^2/\sigma_2^2)$, along with their variances and covariances, are readily available; in addition to this, the user can compute: (i) an estimate for σ_2^2 as $(mn)^{-1}\Pi_1(\bar{\mathbf{x}})$, (ii) the exact log-likelihood at $\bar{\mathbf{x}}$ from $\Pi_1(\bar{\mathbf{x}})$ and $\Pi_2(\bar{\mathbf{x}})$, and (iii) the residuals for the original model (2.2) as $\bar{u}_{1t} = \bar{a}_{1t} - \bar{\omega}_0 \bar{u}_{2t}$, $\bar{u}_{2t} = \bar{a}_{2t}$ [see (2.3)], where $\bar{a}_{1t}$, $\bar{a}_{2t}$ are the residuals for the standard model computed at $\bar{\mathbf{x}}$ and $\bar{\omega}_0$ is the estimate obtained for ω_0.

Currently, this modular design has been implemented by the author using the C programming language, which allows for, among other things, passing any previously coded CAST routine as a formal parameter to module DRIVER. This means that for production purposes, user involvement is reduced to the strict minimum of specifying a standard model and the data in an input file, whereas the possibility of coding more complex and specific CAST routines for EMLE of non-standard models remains open to the user.

4 Checking for Stationarity and Invertibility

Numerical checks for invertibility and stationarity of the resulting standard model (2.1), are needed if one wishes to take advantage of the constraining possibilities offered by the objective function (3.1) (Shea 1984; Mauricio 1995). Adequate numerical checks for stationarity can be found in Shea (1989) and Mauricio (1995, 1996) as byproducts of the computations required for evaluating (3.2); although these checks are necessary but not sufficient in the case of mixed (i.e. ARMA) models, it has never happened in practice that a model satisfying them has turned out to be non-stationary. A numerical check for invertibility can be found in Mauricio (1995, 1996) as a byproduct too; a more conclusive check can be found in Shea (1989), although it requires computing the eigenvalues of a non-symmetric $mq \times mq$ matrix. Numerical checks for stationarity and invertibility can also be found in Luceño (1994); however, they require solving two linear systems of $m(m+1)/2 + m^2(p-1)$ and $m(m+1)/2 + m^2(q-1)$ equations and computing the Cholesky factorizations of two $mp\times mp$ and $mq\times mq$ symmetric matrices, which adds significantly to the overall computational burden of the EMLE process.

5 Concluding Remarks

The techniques outlined in this paper have recently been applied in the development of a new methodology for dealing with seasonal time series (Gallego 1995) as well as in an econometric project on the money supply in Spain (Gonzalo 1995). In addition to that, they are currently being used in both theoretical and applied studies that require efficient and reliable methods for estimating time series models. These studies include econometric projects on the Spanish labour markets and on the Spanish foreign and public sectors as well as methodological projects in the following areas: (i) detection and treatment of influential data, (ii) tests of structural breaks in time series models, and (iii) EMLE of multivariate systems with cointegrated variables.

The author is grateful to Arthur B. Treadway and Víctor M. Gonzalo, whose expert advice has been very valuable in writing this paper, and to the members of the SPS:EE, who have acted as serious testers of the programs originating this paper. Financial support from Caja de Madrid (Spain) is also acknowledged.

References

Ansley, C.F., and Newbold, P. (1980). Finite Sample Properties of Estimators for Autoregressive-Moving Average Models. *Journal of Econometrics*, **13**, 159-183.

Gallego, J.L. (1995). *Una Familia General de Procesos Estocásticos Estacionales.* Tesis Doctoral, Madrid: Universidad Complutense.

Gonzalo, V.M. (1995). *Análisis Econométricos del Proceso de Oferta de Dinero en España, 1964-1990.* Tesis Doctoral, Madrid: Universidad Complutense.

Hillmer, S.C., and Tiao, G.C. (1979). Likelihood Function of Stationary Multiple Autoregressive Moving Average Models. *Journal of the American Statistical Association*, **74**, 652-660.

Luceño, A. (1994). A Fast Algorithm for the Exact Likelihood of Stationary and Partially Nonstationary Vector Autoregressive-Moving Average Processes. *Biometrika*, **81**, 555-565.

Mauricio, J.A. (1995). Exact Maximum Likelihood Estimation of Stationary Vector ARMA Models. *Journal of the American Statistical Association*, **90**, 282-291.

Mauricio, J.A. (1996). ALG 832: The Exact Likelihood Function of a Vector ARMA Model. Forthcoming in *Applied Statistics.*

Shea, B.L. (1984). Maximum Likelihood Estimation of Multivariate ARMA Processes via the Kalman Filter. In *Time Series Analysis: Theory and Practice* (Vol. 5), ed. O.D. Anderson, Amsterdam: North-Holland, pp. 91-101.

Shea, B.L. (1989), Algorithm AS 242: The Exact Likelihood of a Vector Autoregressive-Moving Average Model. *Applied Statistics*, **38**, 161-204.