Statistics and Probability Letters 124 (2017) 92–96

Contents lists available at ScienceDirect

Statistics and Probability Letters

journal homepage: www.elsevier.com/locate/stapro

Computation of vector ARMA autocovariances

Tucker McElroy

Center for Statistical Research and Methodology, U.S. Census Bureau, 4600 Silver Hill Road, Washington, DC 20233-9100, United States

ARTICLE INFO

Article history:
Received 6 December 2016
Received in revised form 27 December 2016
Accepted 29 December 2016
Available online 6 January 2017

Keywords:
Algorithm
Multivariate time series
Wold decomposition

ABSTRACT

This note describes an algorithm for computing the autocovariance sequence of a VARMA process, without requiring the intermediary step of determining the Wold representation. Although the recursive formula for the autocovariances is well-known, the initialization of this recursion in standard treatments (such as Brockwell and Davis (1991) or Lütkepohl (2007)) is slightly nuanced; we provide explicit formulas and algorithms for the initial autocovariances.

Published by Elsevier B.V.

1. Introduction

Consider a vector time series $\{X_t\}$ of dimension m that satisfies an ARMA (p, q) equation

$$\Phi(B)X_t = \Theta(B)\epsilon_t,$$

where $\{\epsilon_t\}$ is vector white noise of covariance Σ, a positive definite $m \times m$ matrix. Here B is the backshift operator, and $\Phi(z)$ and $\Theta(z)$ are respectively order p and q matrix polynomials. For convenience, define $\{W_t\}$ to be the MA(q) process satisfying $W_t = \Theta(B)\epsilon_t$. To set notation,

$$\Phi(z) = 1_m - \sum_{j=1}^{p} \Phi_j z^j \qquad \Theta(z) = 1_m + \sum_{k=1}^{q} \Theta_k z^k,$$

where 1_m is the $m \times m$ identity matrix. Note that $p = 0$ and/or $q = 0$ is allowed. In principle both matrix polynomials are assumed to satisfy the causality/invertibility restrictions, i.e., all complex zeroes z of the determinant of the matrix polynomials lie outside the unit circle.

While much time series literature has focused upon the efficient computation of the Gaussian likelihood – say via the Durbin–Levinson (D–L) or Innovations algorithms – it is also important to be able to speedily compute a model's autocovariances. Numerical optimization of the likelihood typically requires hundreds of function evaluations, and the problem is compounded with higher dimensional data because the parameter space dimension is typically quite large; higher dimensional parameter manifolds require more search directions in optimization techniques, which in turn require many function evaluations. Even a seemingly benign trivariate VARMA(1,1) model involves an 18-dimensional parameter manifold. Of course, VARMA autocovariances are also needed in other applications, such as the computation of projections (forecasts, backcasts, etc.). This paper provides an explicit algorithm for calculating VARMA autocovariances.

For any two vector processes $\{Y_t\}$ and $\{Z_t\}$, the cross-covariance function is defined by $\Gamma_{YZ}(h) = \mathrm{Cov}(Y_t, Z'_{t-h})$ for integer h, and if the processes are jointly weakly stationary then $\Gamma'_{YZ}(-h) = \Gamma_{ZY}(h)$. The VARMA autocovariance function is defined to be $\Gamma_{XX}(h)$, which we only need compute for $h \geq 0$. It is a simple matter to show that Γ_{XX} satisfies $\Phi(B)\Gamma_{XX}(h) = \Gamma_{WX}(h)$,

E-mail address: tucker.s.mcelroy@census.gov.

http://dx.doi.org/10.1016/j.spl.2016.12.015
0167-7152/Published by Elsevier B.V.

where B operates on the lag index h. The quantity $\Gamma_{WX}(h)$ can be expressed in terms of the Wold coefficients (this is the approach given in Brockwell and Davis, 1991, Chapter 11), whereas $\Phi(B)\Gamma_{XX}(h)$ is a recursion. Once the values $\Gamma_{XX}(h)$ for $0 \le h < p$ are known, the recursion will provide all subsequent autocovariances.

Brockwell and Davis (1991) mention that the redundancy in the recursive formula, together with $\Gamma_{XX}(-h) = \Gamma'_{XX}(h)$, can be used to obtain the initial values—this is our approach, described below explicitly. Alternatively, one can embed a VAR(p) or a VARMA(p, q) as a higher-dimensional VAR process that mathematically takes the form of a VAR(1); then the initial p values of $\Gamma_{XX}(h)$ can be extracted from the lag zero autocovariance of the higher-dimensional VAR(1). This is the approach of Lütkepohl (2007).

In other literature, Barone (1987) discusses simulation based on calculating autocovariances from a state space embedding. Mittnik (1990, 1993) gives an explicit formula for the initial lags of the autocovariance function, in terms of the AR and MA coefficients, and explores the use of the D–L algorithm to improve computational efficiency. Our own derivation is mathematically equivalent to Mittnik's approach – which avoids having to determine the Wold representation – but with the individual algorithmic steps broken out.

Our objective is to provide an explicit algorithm that is easily encoded. It is surprisingly difficult to find VARMA autocovariance software on the Internet, written in the R language. Although in principle the above methods (of Lütkepohl, Brockwell and Davis, or Mittnik) can be implemented, it is perhaps not so easy for all but the greatest experts; we here provide an implementation in R, which scientists may find useful to have readily available. In Section 2 we derive the mathematical formulas, followed by some illustrations in Section 3. The R code is supplemental (see Appendix A).

A caution in utilizing the algorithm, is that the AR and MA matrix polynomials are presumed to be stable and invertible, respectively, and the algorithm makes no assurance that such is the case. Passing an unstable AR matrix polynomial to the likelihood will generate an explosive autocovariance sequence; this is a concern during maximum likelihood estimation, wherein proposed values of the AR and MA coefficients may generate unstable matrix polynomials.

2. Derivation of the algorithm

Suppose the processes are mean zero. Multiplying the following equation

$$X_t - \sum_{j=1}^{p} \Phi_j X_{t-j} = W_t$$

on the right by X'_{t-h} and taking expectations yields

$$\Gamma_{XX}(h) = \sum_{j=1}^{p} \Phi_j \Gamma_{XX}(h-j) + \Gamma_{WX}(h). \tag{1}$$

We will apply the vec operator. For short, let vec$(A) = A^\sharp$ for any matrix A. Recall that vec$(AB) = [1_m \otimes A]B^\sharp$. Thus

$$\Gamma^\sharp_{XX}(h) - \sum_{j=1}^{p} [1_m \otimes \Phi_j]\, \Gamma^\sharp_{XX}(h-j) = \Gamma^\sharp_{WX}(h),$$

which holds for all integers h. Consider these equations for $-p \le h \le p$ ($p = 0$ is allowed), and write in block matrix form:

$$\begin{bmatrix} -1_m \otimes \Phi_p & \cdots & 1_m \otimes 1_m & 0 & \cdots \\ 0 & -1_m \otimes \Phi_p & \cdots & 1_m \otimes 1_m & \cdots \\ \ddots & \ddots & \ddots & \ddots & \ddots \\ 0 & \cdots & -1_m \otimes \Phi_p & \cdots & 1_m \otimes 1_m \end{bmatrix} \begin{bmatrix} \Gamma^\sharp_{XX}(-p) \\ \vdots \\ \Gamma^\sharp_{XX}(0) \\ \vdots \\ \Gamma^\sharp_{XX}(p) \end{bmatrix} = \begin{bmatrix} \Gamma^\sharp_{WX}(0) \\ \vdots \\ \Gamma^\sharp_{WX}(p) \end{bmatrix}.$$

Let this large matrix be denoted by A. It is a $(p+1) \times (2p+1)$ block matrix of $m^2 \times m^2$ block matrices. We wish to solve for the vector of Γ_{XX} values, and there is a redundancy to be exploited. Let K denote the commutator matrix such that $\Gamma^\sharp_{XX}(-h) = K\Gamma^\sharp_{XX}(h)$. This matrix K is described in Lütkepohl (2007), and has the property that $K[A \otimes B] = [B \otimes A]K$. As a result, the block matrix product on the left hand side above becomes

$$\begin{bmatrix} -(1_m \otimes \Phi_p)K & \cdots & -(1_m \otimes \Phi_1)K & 1_m \otimes 1_m & 0 & \cdots \\ 0 & -(1_m \otimes \Phi_p)K & \cdots & -1_m \otimes \Phi_1 & 1_m \otimes 1_m & \cdots \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ 0 & \cdots & \cdots & -1_m \otimes \Phi_p & \cdots & 1_m \otimes 1_m \end{bmatrix} \begin{bmatrix} \Gamma^\sharp_{XX}(p) \\ \vdots \\ \Gamma^\sharp_{XX}(0) \\ \vdots \\ \Gamma^\sharp_{XX}(p) \end{bmatrix}.$$

The key difference from the previous equation is that every block matrix in any of the first p block columns of A is now right-multiplied by K, whereas the latter $p+1$ block columns of A are left untouched. Also, the redundancy in the vector of Γ_{XX} values is apparent. This matrix, which we call $\widetilde{A}$, can now be "folded" over to generate $\underline{A}$, as follows: $\underline{A}$ is block $(p+1)\times(p+1)$, with jth block column for $2 \leq j \leq p+1$ given by the sum of the $p+j$th block column of $\widetilde{A}$ summed with the $(p+2-j)$th block column of $\widetilde{A}$. Once the folding is done, the matrix system simplifies and the redundancy in Γ_{XX} disappears, and we obtain

$$\begin{bmatrix} \Gamma^{\sharp}_{XX}(0) \\ \vdots \\ \Gamma^{\sharp}_{XX}(p) \end{bmatrix} = \underline{A}^{-1} \begin{bmatrix} \Gamma^{\sharp}_{WX}(0) \\ \vdots \\ \Gamma^{\sharp}_{WX}(p) \end{bmatrix} \tag{2}$$

so long as this matrix is invertible. This equation holds under the same conditions as the solubility of the high-dimensional VAR(1) representation of a VAR(p) process, as discussed in Lütkepohl (2007); thus, invertibility of $\underline{A}$ is equivalent to stability of the matrix polynomial $\Phi(z)$.

The next step is to compute the cross-covariances Γ_{WX}. Take the equation

$$X'_{t-h} = \sum_{j=1}^{p} X'_{t-h-j}\Phi'_j + W'_{t-h}$$

and multiply it on the left by W_t, and take expectations to obtain

$$\Gamma_{WX}(h) = \sum_{j=1}^{p} \Gamma_{WX}(h+j)\Phi'_j + \Gamma_{WW}(h).$$

This relates the cross-covariance function to the autocovariance function of the moving average portion, which is easy to calculate. Applying the vec operator yields

$$\Gamma^{\sharp}_{WX}(h) = \sum_{j=1}^{p} [\Phi_j \otimes 1_m]\, \Gamma^{\sharp}_{WX}(h+j) + \Gamma^{\sharp}_{WW}(h).$$

Utilizing the causal representation of the time series, it is clear that $\Gamma_{WX}(h) = 0$ for $h > q$. So we consider this equation for $0 \leq h \leq q$, which in matrix form is

$$\begin{bmatrix} 1_m \otimes 1_m & -(\Phi_1 \otimes 1_m) & \cdots & -(\Phi_p \otimes 1_m) \\ 0 & 1_m \otimes 1_m & \cdots & -(\Phi_{p-1} \otimes 1_m) \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & 1_m \otimes 1_m \end{bmatrix} \begin{bmatrix} \Gamma^{\sharp}_{WX}(0) \\ \vdots \\ \Gamma^{\sharp}_{WX}(q) \end{bmatrix} = \begin{bmatrix} \Gamma^{\sharp}_{WW}(0) \\ \vdots \\ \Gamma^{\sharp}_{WW}(q) \end{bmatrix}.$$

Denote this matrix by $\underline{B}$, which is a $(q+1)\times(q+1)$ dimensional block matrix, with blocks of size $m^2 \times m^2$. Thus

$$\begin{bmatrix} \Gamma^{\sharp}_{WX}(0) \\ \vdots \\ \Gamma^{\sharp}_{WX}(q) \end{bmatrix} = \underline{B}^{-1} \begin{bmatrix} \Gamma^{\sharp}_{WW}(0) \\ \vdots \\ \Gamma^{\sharp}_{WW}(q) \end{bmatrix}. \tag{3}$$

Finally, for a vector moving average we have

$$\sum_{|h|\leq q} \Gamma_{WW}(h)B^h = \Theta(B)\Sigma\Theta'(B^{-1}), \tag{4}$$

so that matrix polynomial multiplication yields the $\Gamma_{WW}(h)$ sequence for $0 \leq h \leq q$. The algorithm then consists of the following steps:

1. Compute $\Gamma_{WW}(h)$ for $0 \leq h \leq q$ via (4).
2. Compute $\Gamma_{WX}(h)$ in vectorized form for $0 \leq h \leq q$ via (3).
3. If $p < q$, truncate the $\Gamma_{WX}(h)$ vector so that it only goes up to lag p. But if $p > q$, pad it out with zeroes (if $p = q$, do nothing to it).
4. Compute $\Gamma_{XX}(h)$ in vectorized form for $0 \leq h \leq p$ via (2).
5. Undo the vec operators on Γ_{XX} and Γ_{WX}, and compute $\Gamma_{XX}(h)$ for $h > p$ recursively via (1). When $h > q$, the cross-covariance term makes no contribution.

3. Illustrations

Consider the special case of a VAR(1), where the autocovariance formulas are known (see Lütkepohl, 2007). We verify that our algorithm reduces to this special case. Here $p = 1$ and $q = 0$, so that

$$A = \begin{bmatrix} -1_m \otimes \Phi_1 & 1_m \otimes 1_m & 0 \\ 0 & -1_m \otimes \Phi_1 & 1_m \otimes 1_m \end{bmatrix}.$$

Also

$$\underline{A} = \begin{bmatrix} 1_m \otimes 1_m & -(1_m \otimes \Phi_1)K \\ -(1_m \otimes \Phi_1) & 1_m \otimes 1_m \end{bmatrix} = \begin{bmatrix} K & 0 \\ 0 & 1_m \otimes 1_m \end{bmatrix} \begin{bmatrix} 1_m \otimes 1_m & -(\Phi_1 \otimes 1_m) \\ -(1 \otimes \Phi_1) & 1_m \otimes 1_m \end{bmatrix},$$

and thus the inverse of $\underline{A}$ can be written as the matrix product

$$\underline{A}^{-1} = \begin{bmatrix} 1_m \otimes 1_m & \Phi_1 \otimes 1_m \\ 1_m \otimes \Phi_1 & 1_m \otimes 1_m \end{bmatrix} 1_2 \otimes (1_m \otimes 1_m - \Phi_1 \otimes \Phi_1)^{-1} \begin{bmatrix} K^{-1} & 0 \\ 0 & 1_m \otimes 1_m \end{bmatrix}.$$

Finally, we obtain from (2) – and utilizing (3) and (4) with $q = 0$ –

$$\begin{bmatrix} \Gamma^\sharp_{XX}(0) \\ \Gamma^\sharp_{XX}(1) \end{bmatrix} = \underline{A}^{-1} \begin{bmatrix} \Sigma^\sharp(0) \\ 0 \end{bmatrix} = \begin{bmatrix} (1_m \otimes 1_m - \Phi_1 \otimes \Phi_1)^{-1} \; K^{-1} \; \Sigma^\sharp(0) \\ (1_m \otimes \Phi_1) \; (1_m \otimes 1_m - \Phi_1 \otimes \Phi_1)^{-1} \; K^{-1} \; \Sigma^\sharp(0) \end{bmatrix}.$$

Using the symmetry of Σ, yields $\Gamma_{XX}(1) = \Phi_1 \, \Gamma_{XX}(0)$ and $\Gamma^\sharp_{XX}(0) = (1_m \otimes 1_m - \Phi_1 \otimes \Phi_1)^{-1} \; \Sigma^\sharp$, as desired.

Next, we consider three numerical examples treated in Lütkepohl (2007), corresponding to a VAR(1), a VAR(2), and a VARMA(2,1).

Example 1. We consider the 3-dimensional VAR(1) process given as Example 2.1.14 of Lütkepohl (2007). Here

$$\Phi_1 = \begin{bmatrix} 0.5 & 0 & 0 \\ 0.1 & 0.1 & 0.3 \\ 0 & 0.2 & 0.3 \end{bmatrix} \qquad \Sigma = \begin{bmatrix} 2.25 & 0 & 0 \\ 0 & 1 & 0.5 \\ 0 & 0.5 & 0.74 \end{bmatrix},$$

and the autocovariance function at lags 0 through 3 is given by

$$\Gamma_{XX}(0) = \begin{bmatrix} 3.00000000 & 0.1608833 & 0.01892744 \\ 0.16088328 & 1.1723174 & 0.67368324 \\ 0.01892744 & 0.6736832 & 0.95355460 \end{bmatrix}$$

$$\Gamma_{XX}(1) = \begin{bmatrix} 1.50000000 & 0.08044164 & 0.009463722 \\ 0.32176656 & 0.33542504 & 0.355327448 \\ 0.03785489 & 0.43656845 & 0.420803028 \end{bmatrix}$$

$$\Gamma_{XX}(2) = \begin{bmatrix} 0.75000000 & 0.04022082 & 0.004731861 \\ 0.19353312 & 0.17255720 & 0.162720026 \\ 0.07570978 & 0.19805554 & 0.197306398 \end{bmatrix}$$

$$\Gamma_{XX}(3) = \begin{bmatrix} 0.37500000 & 0.02011041 & 0.002365931 \\ 0.11706625 & 0.08069447 & 0.075937108 \\ 0.06141956 & 0.09392810 & 0.091735925 \end{bmatrix}.$$

These can be compared to the values given in Lütkepohl (2007, page 28).

Example 2. Next, consider the 2-dimensional VAR(2) process given as Example 2.1.15 of Lütkepohl (2007). Here

$$\Phi_1 = \begin{bmatrix} 0.5 & 0.1 \\ 0.4 & 0.5 \end{bmatrix} \qquad \Phi_2 = \begin{bmatrix} 0 & 0 \\ 0.25 & 0 \end{bmatrix} \qquad \Sigma = \begin{bmatrix} 0.09 & 0 \\ 0 & 0.04 \end{bmatrix},$$

and the autocovariance function at lags 0 through 3 is given by

$$\Gamma_{XX}(0) = \begin{bmatrix} 0.13123055 & 0.06609815 \\ 0.06609815 & 0.18130995 \end{bmatrix}$$

$$\Gamma_{XX}(1) = \begin{bmatrix} 0.07222509 & 0.05118007 \\ 0.10359757 & 0.14299363 \end{bmatrix}$$

$$\Gamma_{XX}(2) = \begin{bmatrix} 0.0464723 & 0.0398894 \\ 0.1134965 & 0.1084934 \end{bmatrix}$$

$$\Gamma_{XX}(3) = \begin{bmatrix} 0.03458580 & 0.03079404 \\ 0.09339342 & 0.08299746 \end{bmatrix}.$$

These can be compared to the values given in Lütkepohl (2007, page 29).

Example 3. Next, consider the 2-dimensional VARMA(2,1) process given as Exercise 11.3 of Lütkepohl (2007). The innovation covariance matrix is not specified, so we take it to be the same as the previous example. Thus

$$\Phi_1 = \begin{bmatrix} 0.5 & 0.1 \\ 0.4 & 0.5 \end{bmatrix} \quad \Phi_2 = \begin{bmatrix} 0 & 0 \\ 0.25 & 0 \end{bmatrix} \quad \Theta_1 = \begin{bmatrix} 0.6 & 0.2 \\ 0 & 0.3 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 0.09 & 0 \\ 0 & 0.04 \end{bmatrix},$$

and the autocovariance function at lags 0 through 3 is given by

$$\Gamma_{XX}(0) = \begin{bmatrix} 0.270201 & 0.1908310 \\ 0.190831 & 0.3967657 \end{bmatrix}$$

$$\Gamma_{XX}(1) = \begin{bmatrix} 0.2081836 & 0.1430920 \\ 0.2555418 & 0.3506007 \end{bmatrix}$$

$$\Gamma_{XX}(2) = \begin{bmatrix} 0.1296460 & 0.1066061 \\ 0.2785946 & 0.2802449 \end{bmatrix}$$

$$\Gamma_{XX}(3) = \begin{bmatrix} 0.09268245 & 0.08132754 \\ 0.24320158 & 0.21853790 \end{bmatrix}.$$

The algorithm here is quite similar to that proposed by Lütkepohl (2007), which obtains initial autocovariances by inverting a matrix of dimension $(p+q)m$. The matrices $\underline{A}$ and $\underline{B}$ are not much smaller, and hence in practice we have observed very similar computation times for both algorithms. One minor difference is that the algorithm of Lütkepohl requires the stability of the MA matrix polynomials for the computation of the initial autocovariances, whereas in our algorithm the moving average need not be stable.

Disclaimer This report is released to inform interested parties of research and to encourage discussion. The views expressed on statistical issues are those of the authors and not necessarily those of the U.S. Census Bureau.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at http://dx.doi.org/10.1016/j.spl.2016.12.015.

References

Barone, P., 1987. A method for generating independent realizations of a multivariate normal stationary and invertible ARMA(p, q) process. J. Time Series Anal. 8, 125–130.
Brockwell, P., Davis, R., 1991. Time Series: Theory and Methods. Springer, New York.
Lütkepohl, H., 2007. New Introduction to Multiple Time Series Analysis. Springer-Verlag, Berlin.
Mittnik, S., 1990. Computation of the theoretical autocovariance matrices of multivariate autoregressive moving average time series. J. R. Stat. Soc. Ser. B Stat. Methodol. 52, 151–155.
Mittnik, S., 1993. Computing theoretical autocovariances of multivariate autoregressive moving average models by using a block Levinson method. J. R. Stat. Soc. Ser. B Stat. Methodol. 55, 435–440.