3-2014

Symbolic ARMA Model Analysis

Keith H. Webb

Lawrence Leemis
William & Mary, lmleem@wm.edu

Recommended Citation

Webb, Keith H. and Leemis, Lawrence, Symbolic ARMA Model Analysis (2014). *Computational Economics*, 43(3), 313-330.
10.1007/s10614-013-9373-z

Symbolic ARMA Model Analysis

Keith H. Webb

Lawrence M. Leemis

Department of Mathematics

The College of William & Mary

Williamsburg, VA 23187–8795, USA

khwebb@math.wm.edu

leemis@math.wm.edu

ARMA models provide a parsimonious and flexible mechanism for modeling the evolution of a time series. Some useful measures of these models (e.g., the autocorrelation function or the spectral density function) are tedious to compute by hand. This paper uses a computer algebra system, not simulation, to calculate measures of interest associated with ARMA models.

KEY WORDS: Autocorrelation functions; Computer algebra systems; Spectral density function; Time series analysis; Unit roots analysis.

---

1 Introduction

Many problems in time series analysis rely on approximate values from Monte Carlo simulations or the central limit theorem rather than exact results. The computer algebra system Maple and the APPL (A Probability Programming Language) package can calculate exact

results that would be impractical to compute by hand (Glen, et al. 2001). This paper describes our time series extension to APPL which can compute autocorrelation and partial autocorrelation functions of ARMA (autoregressive moving average) models which would otherwise require extremely tedious pencil and paper work or simulation to find, along with several other tools for working with ARMA models. Simulation is not used in any of the results. [1]

The time series extension to APPL provides procedures for ARMA models to:

- calculate an autocorrelation (`TSCorrelation`), including autocorrelations of models with heteroskedastic error terms,
- calculate a partial autocorrelation (`TSPartialCorrelation`),
- calculate a mean (`TSMean`),
- calculate a variance or covariance (`TSVariance`),
- plot an autocorrelation or partial correlation function (`TSPlot`),
- calculate a spectral density function (`SDF`),
- perform a unit roots analysis to determine whether or not an AR(p) model is stationary or an MA(q) model is invertible (`UnitRoots`),
- forecast an AR(p) model and calculate a confidence interval (`TSForecast`),
- generate a realization of a process (`TSRealization`),
- perform an exploratory time series analysis (`ETSA`) which displays several of the previous commands at once.

[1] The extension can be downloaded from `http://www.math.wm.edu/~leemis/TSAPPL.txt` and APPL can be downloaded from `http://applsoftware.com/`

Section 2 describes the data structure for storing ARMA models and explains the ways that the mean, variance, autocorrelation, partial autocorrelation and forecasts are computed. Section 3 illustrates how to use the time series extension through a series of examples.

1.1 Basics of ARMA Models and APPL

An AR(p) model has p autoregressive components, 0 moving average components and is defined by

$$Y_t = c + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} + \epsilon_t, \tag{1}$$

where Y_t is the value of the time series at time t, c is a real-valued constant, $\phi_1, \phi_2, \ldots, \phi_p$ are real-valued parameters, and $\epsilon_0, \epsilon_1, \epsilon_2, \ldots, \epsilon_t$ are error terms, typically mutually independent and normally distributed with a mean of 0. An MA(q) model has 0 autoregressive components, q moving average components and is defined by

$$Y_t = c + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q}, \tag{2}$$

with variables defined similarly. An ARMA(p, q) model is a combination of (1) and (2):

$$Y_t = c + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q}. \tag{3}$$

One aspect of analyzing ARMA (and other time series) models is finding autocorrelation functions, that is, the j^{th} autocorrelation, denoted by ρ_j, is the correlation between the value of a time series Y_t at time t and its value Y_{t-j} at time $t - j$. For models where the ϵ_t terms are independent and identically distributed (IID), ρ_j is the same at any time t. The j^{th} autocorrelation of MA(q) models with $q < \infty$ drops off to 0 when $j > q$ because Y_t and Y_{t-j} do not share terms. On the other hand, the j^{th} autocorrelation of stationary AR(p) models asymptotically approaches 0 as j increases because the value of Y_t affects all future

values, but its impact decreases over time. Finding the exact autocorrelation function for an ARMA model can be difficult to do by hand, especially when error terms are not IID. Section 2 explains two ways that this process is automated with Maple and APPL.

APPL is essential for working with various distributions of error terms. For example, suppose that $\epsilon_t \sim$ exponential(4) and we want to find $E[\epsilon_t^2]$. APPL can find the exact solution quickly:

```
> X := ExponentialRV(4);
> g := [[x -> x ^ 2], [0, infinity]];
> U := Transform(X, g);
> Mean(U);
```

$$\frac{1}{8}$$

The `Mean`, `Variance`, and `Transform` procedures, as well as the data structure for probability distributions in APPL, are all used when working with ARMA models. This enables our software to use an arbitrary probability distribution for the error terms. For more on APPL, see Glen et al. (2001).

2 Implementation

The time series analysis extension to the APPL software consists of approximately 1000 lines of Maple code which allow a user to define an ARMA model and compute various measures associated with the model. The data structure for storing the ARMA model and methods for computing the autocorrelation and partial autocorrelation are described here.

2.1 Data Structure

One of the first steps in developing the time series extension is to define a data structure within Maple to store a times series model. A list-of-lists data structure was used to describe an ARMA model. This allows for both symbolic and numeric values for the parameters, and the error terms ϵ_t can have arbitrary probability distributions. The first sub-list contains the parameters $\phi_1, \phi_2, \ldots, \phi_p$ for the autoregressive portion of the model; the second sub-list contains the parameters $\theta_1, \theta_2, \ldots, \theta_q$ for the moving average portion of the model; the third element of the list contains the probability distribution of the error terms. For example, the statement

```
> X := [[0.2, 0.93], [1.4], NormalRV(0, sigma)];
```

sets the variable `X` to an ARMA(2, 1) model with $\phi_1 = 0.2$, $\phi_2 = 0.93$, $\theta_1 = 1.4$, and $N(0, \sigma^2)$ error terms (Gaussian white noise). Procedures `AR`, `MA`, and `ARMA` have also been written to save on keystrokes. For example,

```
> Y := ARMA(1, 1);
```

sets the variable `Y` to a ARMA(1, 1) model with arbitrary parameters ϕ_1, θ_1, and the default standard normal error terms (the default allows a realization to be computed). Other error terms can also be included using these procedures, for example,

```
> X := MA([2, 3, 1 / 4], ExponentialRV(1));
```

sets the variable `X` to an MA(3) model with parameters $\theta_1 = 2$, $\theta_2 = 3$, $\theta_3 = 1/4$, and unit exponential error terms. Time-dependent error terms can be introduced in the data structure by using the variable `tau`, the time series placeholder for t in the expression for the error-term distribution. Time-dependent errors are illustrated in Section 3.

2.2 Computing Autocorrelation by the Defining Formula

The autocorrelation between Y_t and Y_{t-j} can be calculated using the defining formula:

$$\rho_j = \frac{\mathrm{Cov}(Y_t, Y_{t-j})}{\sigma_{Y_t}\sigma_{Y_{t-j}}} = \frac{E[(Y_t - E[Y_t])(Y_{t-j} - E[Y_{t-j}])]}{\sqrt{E[(Y_t - E[Y_t])^2]}\sqrt{E[(Y_{t-j} - E[Y_{t-j}])^2]}}. \tag{4}$$

For MA(q) models, this is a straightforward but tedious computation to perform by hand. Since for any random variables X_1, X_2 and constants a, b we have

$$E[aX_1 + bX_2] = aE[X_1] + bE[X_2], \tag{5}$$

then

$$\begin{aligned} E[Y_t] &= E[c + \epsilon_t + \theta_1\epsilon_{t-1} + \theta_2\epsilon_{t-2} + \cdots + \theta_q\epsilon_{t-q}] \\ &= c + E[\epsilon_t] + \theta_1 E[\epsilon_{t-1}] + \theta_2 E[\epsilon_{t-2}] + \cdots + \theta_q E[\epsilon_{t-q}]. \end{aligned} \tag{6}$$

Maple and APPL can then compute $E[Y_t]$ term-by-term, even for error term distributions that are non-normal or time-dependent (i.e., not IID). The computed values of $E[Y_t]$ and $E[Y_{t-j}]$ are then substituted into (4). The resulting expressions within each of the expected values are then expanded so that Maple and APPL can compute each expected value term-by-term once again using (5). With these results, Maple can compute the mean, variance, j^{th} covariance and j^{th} autocorrelation for MA(q) models.

For stationary AR(p) and ARMA(p, q) models, this method will not work because computing $E[Y_t]$ requires computing $E[Y_{t-1}]$, which in turn requires computing $E[Y_{t-2}]$, etc. However, autocorrelations for either stationary or non-stationary AR(p) and ARMA(p, q) models over a finite time horizon can be computed using this method. For example, consider

an AR(2) model at time 7,

$$Y_7 = c + \phi_1 Y_6 + \phi_2 Y_5 + \epsilon_7 \tag{7}$$

and let the two initial observations be $Y_0 = c + \epsilon_0$ and $Y_1 = c + \phi_1 Y_0 + \epsilon_1$. Substitute $c + \phi_1 Y_5 + \phi_2 Y_4 + \epsilon_6$ for Y_6 in (7), then substitute $c + \phi_1 Y_4 + \phi_2 Y_3 + \epsilon_5$ for $Y_5, \ldots, c + \epsilon_0$ for Y_0. The resulting Y_7 will not contain any Y_t terms and thus the j^{th} autocorrelation of Y_7, $\rho_{j,7}$, can be computed using the method described above.

2.3 Computing Autocorrelation with the Yule–Walker Equations

Autocorrelation for a steady-state AR(p) model (i.e., a stationary process that has run for long enough that covariances are equal) cannot be computed using the above method. Instead, a well-known system of equations called the Yule–Walker equations describes the autocorrelations (provided that the error terms are IID). For an AR(p) model, the Yule–Walker equations are (Hamilton 1994, page 59):

$$\begin{aligned}
\rho_0 &= 1 \\
\rho_1 &= \phi_1 \rho_0 + \phi_2 \rho_{-1} + \cdots + \phi_p \rho_{1-p} \\
\rho_2 &= \phi_1 \rho_1 + \phi_2 \rho_0 + \cdots + \phi_p \rho_{2-p} \\
&\vdots \\
\rho_p &= \phi_1 \rho_{p-1} + \phi_2 \rho_{p-2} + \cdots + \phi_p \rho_0.
\end{aligned} \tag{8}$$

Since for real–valued processes $\rho_1 = \rho_{-1}, \rho_2 = \rho_{-2}, \ldots$, we can solve for any ρ_j whenever the process is stationary. Covariances can be computed with the similar system of equations,

where γ_j is the j^{th} covariance:

$$\begin{aligned}
\gamma_0 &= \phi_1\gamma_1 + \phi_2\gamma_2 + \cdots + \phi_p\gamma_p + \sigma^2 \\
\gamma_j &= \phi_{j-1}\gamma_1 + \phi_{j-2}\gamma_2 + \cdots + \phi_p\gamma_{j-p} \qquad\qquad j = 1, 2, \ldots . \qquad (9)
\end{aligned}$$

For a stationary ARMA(p, q) process there is not an easily definable system of equations for computing autocorrelations. However, a system can be generated in the following manner. Consider finding the equation for γ_j of an ARMA(p, q) model, $j \leq q$. Observe that when $j > q$, then (9) describes γ_j. For simplicity, let $c = 0$ even though the results hold when this is not the case.

$$\begin{aligned}
Y_t &= \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} + \epsilon_t + \theta_1\epsilon_{t-1} + \theta_2\epsilon_{t-2} + \cdots + \theta_q\epsilon_{t-q} \\
E[Y_t Y_{t-j}] &= E[Y_{t-j}(\phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} + \epsilon_t + \theta_1\epsilon_{t-1} + \theta_2\epsilon_{t-2} + \cdots + \theta_q\epsilon_{t-q})] \\
\gamma_j &= \phi_1\gamma_{j-1} + \phi_2\gamma_{j-2} + \cdots + \phi_p\gamma_{j-p} + E[Y_{t-j}(\epsilon_t + \theta_1\epsilon_{t-1} + \theta_2\epsilon_{t-2} + \cdots + \theta_q\epsilon_{t-q})].
\end{aligned}$$

Now define $\xi_k = E[Y_t\epsilon_{t-k}]$ and let $\theta_0 = 1$, giving us the following solvable system of equations (once the expectations ξ_k are computed):

$$\begin{aligned}
\gamma_j &= \phi_1\gamma_{j-1} + \phi_2\gamma_{j-2} + \cdots + \phi_p\gamma_{j-p} + \theta_j\xi_0 + \theta_{j+1}\xi_1 + \cdots + \theta_q\xi_{q-j} \qquad j \leq q \\
\gamma_j &= \phi_{j-1}\gamma_1 + \phi_{j-2}\gamma_2 + \cdots + \phi_p\gamma_{j-p} \qquad\qquad j > q. \qquad (10)
\end{aligned}$$

The expectation ξ_k can be computed in the following way and then substituted into (10):

$$\begin{aligned}
\xi_0 &= E[\epsilon_t(\phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} + \epsilon_t + \theta_1\epsilon_{t-1} + \theta_2\epsilon_{t-2} + \cdots + \theta_q\epsilon_{t-q})] \\
&= E[\epsilon_t^2] = \sigma^2 \\
\xi_1 &= E[\epsilon_{t-1}(\phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} + \epsilon_t + \theta_1\epsilon_{t-1} + \theta_2\epsilon_{t-2} + \cdots + \theta_q\epsilon_{t-q})] \\
&= \phi_1\xi_0 + \theta_1\sigma^2 \\
\xi_2 &= E[\epsilon_{t-2}(\phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} + \epsilon_t + \theta_1\epsilon_{t-1} + \theta_2\epsilon_{t-2} + \cdots + \theta_q\epsilon_{t-q})] \\
&= \phi_1\xi_1 + \phi_2\xi_0 + \theta_2\sigma^2 \\
&\vdots \\
\xi_k &= E[\epsilon_{t-k}(\phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} + \epsilon_t + \theta_1\epsilon_{t-1} + \theta_2\epsilon_{t-2} + \cdots + \theta_q\epsilon_{t-q})] \\
&= \phi_1\xi_{k-1} + \phi_2\xi_{k-2} + \cdots + \phi_k\xi_0 + \theta_k\sigma^2.
\end{aligned}$$

Maple will solve (8), (9) and (10) either with arbitrary parameters or with values that describe a stationary process.

2.4 Computing Partial Autocorrelation

Partial autocorrelations are computed by first finding autocorrelations with one of the above methods. Then the j^{th} partial autocorrelation is (Woodward and Gray, 1981, pages 579–580)

$$\alpha_j = \frac{\begin{vmatrix} \rho_0 & \rho_1 & \rho_2 & \cdots & \rho_{j-2} & \rho_1 \\ \rho_1 & \rho_0 & \rho_1 & \cdots & \rho_{j-3} & \rho_2 \\ \rho_2 & \rho_1 & \rho_0 & \cdots & \rho_{j-4} & \rho_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \rho_{j-1} & \rho_{j-2} & \rho_{j-3} & \cdots & \rho_1 & \rho_j \end{vmatrix}}{\begin{vmatrix} \rho_0 & \rho_1 & \rho_2 & \cdots & \rho_{j-2} & \rho_{j-1} \\ \rho_1 & \rho_0 & \rho_1 & \cdots & \rho_{j-3} & \rho_{j-2} \\ \rho_2 & \rho_1 & \rho_0 & \cdots & \rho_{j-4} & \rho_{j-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \rho_{j-1} & \rho_{j-2} & \rho_{j-3} & \cdots & \rho_1 & \rho_0 \end{vmatrix}}, \tag{11}$$

which can be evaluated using Maple's `LinearAlgebra` package.

2.5 Invertibility for MA(q) Models

MA(q) models as defined by (2) are invertible if they can be represented by an AR(∞) model. For models with IID error terms, this is true when the roots of the model's characteristic equation lie outside the unit circle in the complex plane. The characteristic equation of an MA(q) model is

$$1 + \theta_1 z + \theta_2 z^2 + \cdots + \theta_q z^q = 0. \tag{12}$$

However, non-invertible MA(q) models have an invertible representation with identical moments (Hamilton 1994, page 68). If $\lambda_1, \lambda_2, \ldots, \lambda_n$ are the roots of (12) inside the unit circle and $\lambda_{n+1}, \lambda_{n+2}, \ldots, \lambda_q$ are the roots of (12) outside the unit circle, then an invertible

model with identical moments is

$$Y_t = c + \left(\prod_{i=1}^{n}(1-\lambda_i L)\right)\left(\prod_{i=n+1}^{q}(1-\lambda_i^{-1}L)\right)\epsilon_t', \tag{13}$$

where $V[\epsilon_t'] = \sigma^2\lambda_1^{-2}\lambda_2^{-2}\cdots\lambda_n^{-2}$ and L is the lag or backshift operator, i.e., $\epsilon_t L = \epsilon_{t-1}$, $\epsilon_t L^2 = \epsilon_{t-2}$, etc.

2.6 Forecasting AR(p) Models

Consider an AR(p) model where the values of $Y_t, Y_{t-1}, \ldots$ are known, but the values of $Y_{t+1}, Y_{t+2}, \ldots$ are unknown, and we want to find the expected value and a confidence interval for Y_{t+s} for some positive integer s. Then,

$$\begin{aligned}
Y_{t+s} &= \phi_1 Y_{t+s-1} + \phi_2 Y_{t+s-2} + \cdots + \phi_p Y_{t+s-p} + \epsilon_{t+s} \\
Y_{t+s-1} &= \phi_1 Y_{t+s-2} + \phi_2 Y_{t+s-3} + \cdots + \phi_p Y_{t+s-p-1} + \epsilon_{t+s-1} \\
&\vdots \\
Y_{t+1} &= \phi_1 Y_t + \phi_2 Y_{t-1} + \cdots + \phi_p Y_{t-p+1} + \epsilon_{t+1}.
\end{aligned}$$

Since Y_{t+1} is an error term plus a constant, then we can substitute Y_{t+1} into Y_{t+2}, then substitute Y_{t+2} into $Y_{t+3}, \ldots$, then substitute Y_{t+s-1} into Y_{t+s}, resulting in an expression consisting of error terms and constants. The software assumes that the error terms are mutually independent and normally distributed, which implies that the value of Y_{t+s} is also normally distributed. Maple can then compute the mean and variance of Y_{t+s} and call APPL's `IDF` procedure to find values for a confidence interval.

3 Examples

This section consists of a series of examples that highlight the capability of the time series extension to the APPL language. Some of the long symbolic results may find use in statistical software which currently use simulation to find measures of interest.

Example 1: MA(2) model with time-dependent error terms. Consider the MA(2) model, $Y_t = c + \epsilon_t + \theta_1\epsilon_{t-1} + \theta_2\epsilon_{t-2}$, where $\epsilon_t \sim N(0, \ln(t)^2), \epsilon_{t-1} \sim N(0, \ln(t-1)^2)$, etc. First we will compute the second autocorrelation at time t, $\rho_{2,t}$, by hand and then we will compute $\rho_{2,t}$ using the time series extension.

Using the defining formula for correlation, we have

$$\rho_{2,t} = \frac{\text{Cov}(Y_t, Y_{t-2})}{\sigma_{Y_t}\sigma_{Y_{t-2}}} = \frac{E[(Y_t - E[Y_t])(Y_{t-2} - E[Y_{t-2}])]}{\sqrt{E[(Y_t - E[Y_t])^2]}\sqrt{E[(Y_{t-2} - E[Y_{t-2}])^2]}}.$$

Notice that since all error terms have a mean of 0, then $E[Y_t] = c$ and $E[Y_{t-2}] = c$. Substituting in the expressions for Y_t and Y_{t-2} we have,

$$\rho_{2,t} = \frac{E[(\epsilon_t + \theta_1\epsilon_{t-1} + \theta_2\epsilon_{t-2})(\epsilon_{t-2} + \theta_1\epsilon_{t-3} + \theta_2\epsilon_{t-4})]}{\sqrt{E[(\epsilon_t + \theta_1\epsilon_{t-1} + \theta_2\epsilon_{t-2})^2]}\sqrt{E[(\epsilon_{t-2} + \theta_1\epsilon_{t-3} + \theta_2\epsilon_{t-4})^2]}}.$$

While the error term distributions are not identical, they are independent and have mean 0, meaning that for any $i \neq j$ we have $E[\epsilon_{t-i}\epsilon_{t-j}] = 0$. This allows us to simplify when expanding the polynomials inside the expected values. So we have,

$$\rho_{2,t} = \frac{E[\theta_2\epsilon_{t-2}^2]}{\sqrt{E[\epsilon_t^2 + \theta_1^2\epsilon_{t-1}^2 + \theta_2^2\epsilon_{t-2}^2]}\sqrt{E[\epsilon_{t-2}^2 + \theta_1^2\epsilon_{t-3}^2 + \theta_2^2\epsilon_{t-4}^2]}}.$$

Each θ_i parameter is a constant (i.e., we can bring them outside of the expected value operator), so it remains to find $E[\epsilon_t^2], \ldots, E[\epsilon_{t-4}^2]$ to complete our calculation of $\rho_{2,t}$. For any $k > 0$ and $\epsilon_k \sim N(0, \ln(k)^2)$ we can use the properties of the normal and chi-square

distributions to find $E[\epsilon_k^2]$. First, $\epsilon_k^2/\ln(k)^2 \sim \chi^2(1)$, which implies that $E[\epsilon_k^2/\ln(k)^2] = 1$ because the expected value of a chi-square random variable is its degrees of freedom. Then, since $1/\ln(k)^2$ is a constant, we can bring it outside of the expected value operator and solve, giving us $E[\epsilon_k^2] = \ln(k)^2$. Using this result, we have:

$$\rho_{2,t} = \frac{\theta_2 \ln(t-2)^2}{\sqrt{\ln(t)^2 + \theta_1^2 \ln(t-1)^2 + \theta_2^2 \ln(t-2)^2}\sqrt{\ln(t-2)^2 + \theta_1^2 \ln(t-3)^2 + \theta_2^2 \ln(t-4)^2}}.$$

Now we will compute the same autocorrelation in just two commands using the time series extension.

```
> X := MA(2, NormalRV(0, ln(tau))):
> TSCorrelation(X, 2);
```

$$\frac{\theta_2 \ln(\mathtt{t}-2)^2}{\sqrt{\ln(\mathtt{t})^2 + \theta_1^2 \ln(\mathtt{t}-1)^2 + \theta_2^2 \ln(\mathtt{t}-2)^2}\sqrt{\ln(\mathtt{t}-2)^2 + \theta_1^2 \ln(\mathtt{t}-3)^2 + \theta_2^2 \ln(\mathtt{t}-4)^2}}$$

The first command assigns the time series list-of-lists data structure to `X`. The first argument of the `MA()` procedure is the order of the model and the second argument is the distribution of the error terms in APPL format (APPL takes standard deviation rather than variance for its probability distributions), with `tau` as the placeholder for t. The second command computes the second autocorrelation of the assigned model.

Example 2: Unit Roots Analysis of an MA(3) model. Find the unit roots of an MA(3) model with parameters $\theta_1 = 0.7$, $\theta_2 = 0.9$, and $\theta_3 = -1.3$. Since all MA(q) models are stationary, a unit root analysis of an MA(q) model tests for invertibility. If the model is not invertible, then an invertible representation will be found according to equation (13) in Section 2.5. A call to the procedure `UnitRoots` prints the values of the unit roots, plots the unit roots in the complex plane, and gives an invertible representation of the model with identical autocorrelation function.

```
> X := MA([0.7, 0.9, -1.3]):
> UnitRoots(X):
```

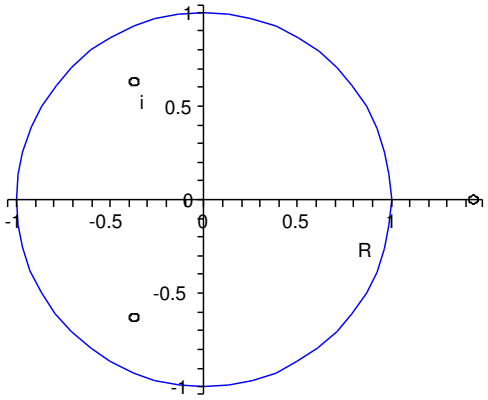
The roots are:

$$[-0.3730587892 + 0.6289673928\mathrm{I}, -0.3730587892 - 0.6289673928\mathrm{I}, 1.438425271]$$

Number of roots inside the unit circle: 2

Assuming error terms are i.i.d, then an invertible representation is:

$$\mathrm{Y_t} = \mathrm{c} + \epsilon_\mathrm{t} + 0.0509128847\epsilon_{\mathrm{t}-1} + 0.0160683988\epsilon_{\mathrm{t}-2} - 0.3717765894\epsilon_{\mathrm{t}-3}$$



If the parameter values were given as exact values rather than decimals (and $q \leq 4$), then the invertible model would also have exact parameters. So if the model were instead input as

```
> X := MA([7 / 10, 9 / 10, -13 / 10]):
```

then the `UnitRoots` procedure prints the exact values of the three roots of the cubic polynomial

$$1 + \frac{7}{10}z + \frac{9}{10}z^2 - \frac{13}{10}z^3 = 0,$$

which are the real root

$$\frac{1}{78}\sqrt[3]{217836+780\sqrt{73329}}+\frac{236}{13\sqrt[3]{217836+780\sqrt{73329}}}+\frac{3}{13}$$

and the complex conjugates

$$-\frac{1}{156}\sqrt[3]{217836+780\sqrt{73329}}-\frac{118}{13\sqrt[3]{217836+780\sqrt{73329}}}+\frac{3}{13}\pm\frac{i\sqrt{3}}{2}\left(\frac{1}{78}\sqrt[3]{217836+780\sqrt{73329}}-\frac{236}{13\sqrt[3]{217836+780\sqrt{73329}}}\right).$$

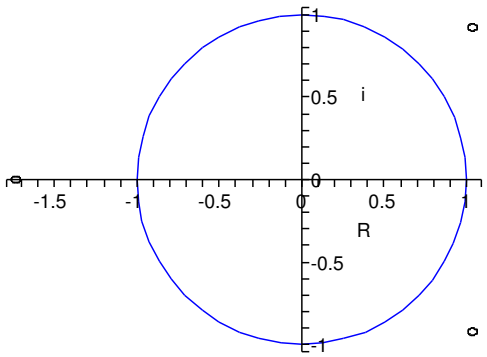
Example 3: Unit Roots Analysis of an AR(3) model. Instead of testing for invertibility as in the previous example, unit roots analysis of AR(p) and ARMA(p, q) models tests for stationarity. AR(p) models with IID error terms are stationary when the roots of $1-\phi_1 z-\phi_2 z^2-\cdots-\phi_p z^p=0$ lie outside the unit circle.

```
> X := AR([0.5, 0.1, -0.3]):
> UnitRoots(X);
```

The roots are:

$$[1.034099254+0.9230480107I,\ 1.034099254-0.9230480107I,\ -1.734865174]$$

All roots lie outside the unit circle, so the ARMA process is stationary



Example 4: MA(1) model with exponential error terms. As expected, non-normal but IID error terms produce the same autocorrelation function as the case with normal error terms.

```
> X := MA(1, ExponentialRV(4)):
> TSCorrelation(X, 1);
```

$$\frac{\theta_1}{1+\theta_1^2}$$

Example 5: AR(2) model at time 7. In this example, $\rho_{2,7}$ for $Y_7 = c + \phi_1 Y_6 + \phi_2 Y_5 + \epsilon_7$ of an AR(2) model is calculated using recursive substitution then taking of expected values as described in Section 2.2. Notice that the constant term, c, appears in the expression.

```
> X := AR([1 / 2, 1 / 4]):
> TSCorrelation(X, 2, 7);
```

$$\frac{1067\sqrt{30621}\sqrt{16384}}{31355904\sqrt{\frac{2209}{16384}\,c^2+\frac{461}{256}}}$$

This can also be evaluated with arbitrary parameters, but the resulting expression for the 2^{nd} autocorrelation value is very long.

Example 6: AR(4) model with autocorrelation computed by the Yule–Walker equations. We can compute ρ_2 for an AR(4) model that is in steady-state (i.e., stationary) by the Yule–Walker method described in Section 2.3. The optional `"YW"` flag tells Maple to find the correlation by solving the Yule–Walker equations rather than the method described in Section 2.2.

```
> X := AR(4):
> TSCorrelation(X, 2, "YW");
```

$$-\frac{-\phi_4{}^2\phi_2+\phi_4\phi_3{}^2+\phi_1\phi_4\phi_3-\phi_2{}^2\phi_4+\phi_1{}^2+\phi_2+\phi_1\phi_3-\phi_2{}^2}{-1+\phi_4+\phi_2+\phi_4{}^2-\phi_4{}^3-\phi_4{}^2\phi_2+\phi_1\phi_3+\phi_3{}^2+\phi_1{}^2\phi_4+\phi_1\phi_4\phi_3}$$

Example 7: ARMA(3, 3) model at time 8. The 2nd autocorrelation of Y_8 is computed as described in Section 2.2.

```
> X := ARMA([1 / 2, 3 / 10, 1 / 10], [9 / 10, 1 / 2, 3 / 10]):
> TSCorrelation(X, 2, 8);
```

$$\frac{3935943161}{31174413101120000000}\sqrt{97420040941}\sqrt{6400000000}\frac{1}{\sqrt{\frac{204404181}{16000000}+\frac{14874485521}{25600000000}c^2}}$$

This can be computed with arbitrary parameters but the resulting equation is very long. Again, notice that the constant term, c, appears in the expression.

Example 8: ARMA(3, 2) autocorrelation by solving the Yule–Walker equations. The 2nd autocorrelation of Y_t is computed as described in Section 2.3.

```
> X := ARMA([5 / 10, 3 / 10, 1 / 10], [3 / 4, 1 / 4]):
> TSCorrelation(X, 2, "YW");
```

$$\frac{1636}{1831}$$

Example 9: MA(2) partial autocorrelation. The 3rd partial autocorrelation of an MA(2) model is computed as described in Section 2.4.

```
> TSPartialCorrelation(MA(2), 3);
```

$$\frac{\theta_1\,(1+\theta_2)\left(-2\,\theta_2-2\,{\theta_2}^3+{\theta_1}^2+{\theta_2}^2{\theta_1}^2+{\theta_2}^2\right)}{1+2\,{\theta_2}^2+{\theta_1}^6+{\theta_1}^2+{\theta_1}^4+2\,{\theta_2}^4-2\,\theta_2{\theta_1}^2+{\theta_2}^2{\theta_1}^4+5\,{\theta_2}^2{\theta_1}^2-4\,\theta_2{\theta_1}^4-2\,{\theta_2}^3{\theta_1}^2+{\theta_2}^6+{\theta_2}^4{\theta_1}^2}$$

Example 10: MA(1) partial autocorrelation with time–dependent error terms. The 3rd partial autocorrelation is computed as described in Section 2.4 and $\epsilon_t \sim N(0,t)$.

```
X := MA(1, NormalRV(0, sqrt(tau))):
TSPartialCorrelation(X, 3);
```

$$\frac{{\theta_1}^3\,(\mathtt{t}-1)^3}{\sqrt{\mathtt{t}+{\theta_1}^2\mathtt{t}-{\theta_1}^2}\sqrt{\mathtt{t}-1+{\theta_1}^2\mathtt{t}-2\,{\theta_1}^2}\left(\mathtt{t}^2-\mathtt{t}+{\theta_1}^4\mathtt{t}^2-3\,{\theta_1}^4\mathtt{t}-{\theta_1}^2+2\,{\theta_1}^4\right)}$$

Example 11: AR(3) partial autocorrelation at time 7. In this example, the first two autocorrelations of Y_7 are found using the method described in Section 2.2 which are then used to find the 2nd partial autocorrelation as described in Section 2.4.

```
> TSPartialCorrelation(AR([1 / 2, 3 / 10, 1 / 10]), 2, 7);
```

$$\frac{1}{132731033}\left(\frac{22811879108847780543893}{33554432000000000000000}\,\mathtt{c}^2-\frac{58277673273951481}{83886080000000000000000}\,\sqrt{132731033}\sqrt{51200000}\right.$$
$$\left.\sqrt{\frac{170119849}{256000000}\,\mathtt{c}^2+\frac{289249}{128000}}+\frac{89625931935942986179113}{838860800000000000000}\right)\sqrt{132731033}\sqrt{51200000}$$
$$\frac{1}{\sqrt{\frac{170119849}{256000000}\,\mathtt{c}^2+\frac{289249}{128000}}}\left(\frac{5269767243516833}{13107200000000000}\,\mathtt{c}^2+\frac{2761515025601}{1000000000000}\right)^{-1}$$

Again, this partial autocorrelation can be computed with arbitrary parameters but the resulting expression is very long. As in two of the above examples, the constant term, c,

appears in the expression.

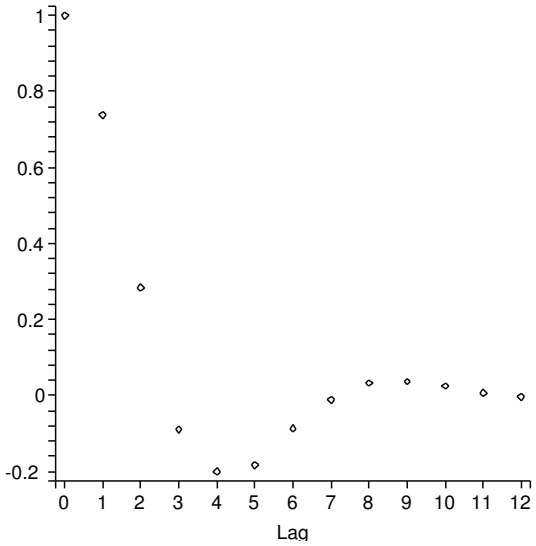
Example 12: AR(3) partial autocorrelation by solving the Yule–Walker equations. By using the `"YW"` flag in `TSPartialCorrelation`, the computer finds the autocorrelations by Yule–Walker which it then uses to find the 2nd partial autocorrelation as described in Section 2.4.

```
> TSPartialCorrelation(AR(3), 2, "YW");
```

$$-\frac{4\,\phi_3\phi_1\phi_2+\phi_2{\phi_3}^2-\phi_2+2\,{\phi_2}^2-\phi_1\phi_2{}^2\phi_3-{\phi_2}^3+{\phi_1}^3\phi_3+2\,{\phi_1}^2{\phi_3}^2+{\phi_1}^2\phi_2+{\phi_3}^3\phi_1-\phi_3\phi_1}{{\phi_1}^2{\phi_3}^2+2\,{\phi_3}^3\phi_1-2\,\phi_3\phi_1+{\phi_3}^4-2\,{\phi_3}^2+2\,\phi_2{\phi_3}^2+1-2\,\phi_2+{\phi_2}^2-{\phi_2}^2{\phi_3}^2-{\phi_1}^2}$$

Example 13: Plotting ARMA(3, 3) autocorrelation. `TSPlot` will find several autocorrelations or partial autocorrelations (by using the optional `"partial"` flag as a third parameter) and then plot them. In this example, the `"YW"` flag is used to tell Maple to find autocorrelations by solving the Yule–Walker equations as described in Section 2.3.

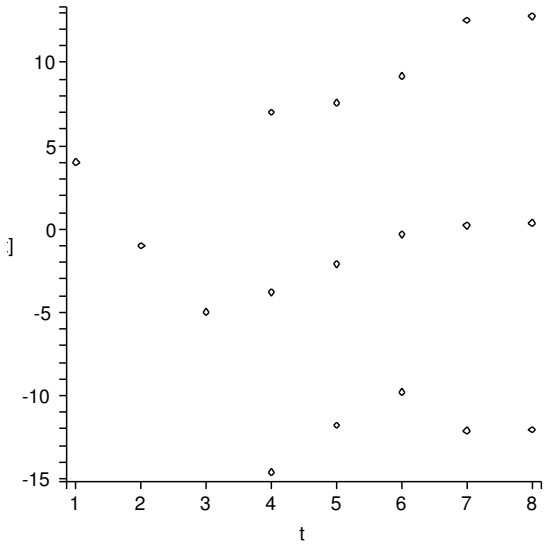
```
> X := ARMA([0.4, 0.2, -0.3], [0.9, 0.3, -0.2]):
> TSPlot(X, "YW");
```



Example 14: Forecasting an AR(3) model. In this example an AR(3) model is forecast 5 time periods into the future as described in Section 2.6. The last three known values of the model are `[-5, -1, 4]` , with `-5` being the most recent. Such small data sets may be used, for example, for the price of a stock shortly after its launch or for space missions where there is limited data on past missions. The fourth (optional) argument in `TSForecast` contains the lower and upper percentiles on the confidence interval. The default is a symmetric 90% confidence level. This example plots symmetric 95% confidence intervals.

```
> X := AR([0.5, 0.1, -0.3], NormalRV(0, 4)):
> TSForecast(X, 5, [-5, -1, 4], [.025, .975]);
```

Assuming error terms are normal for this forecast.



Expected value and confidence interval are:

$$[0.36950, 12.78829339, -12.04929337]$$

After the three known values are plotted, the next five expected values and confidence interval bounds are plotted.

Example 15: ARMA(3, 3) realization. `TSRealization` will sample normally distributed error terms with the Box–Muller transformation and then plot a realization for the given model. If AR parameters are given, then the system warms up before plotting. This example plots a realization of 75 values (the default is 50 values).

```
> X := ARMA([5 / 10, 3 / 10, 1 / 10], [1 / 10, 5 / 10, 7 / 10]):
> TSRealization(X, 75);
```

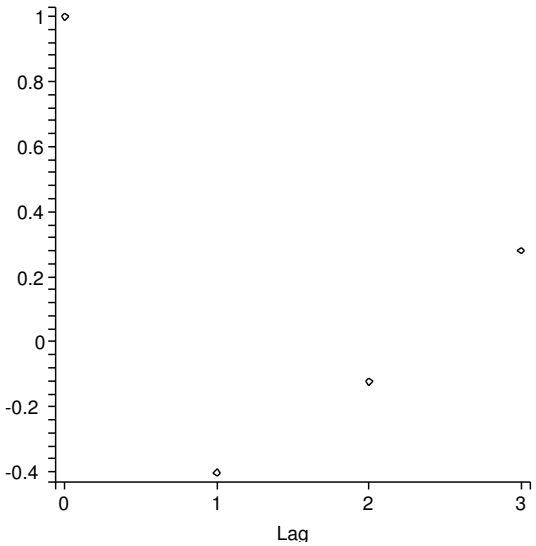
Example 16: Exploratory time series analysis of an MA(3) model. The `ETSA()` command calls autocorrelation, partial autocorrelation, unit roots analysis, spectral density function, and realization procedures.

```
> X := MA([1 / 5, -7 / 5, 11 / 5]):
> ETSA(X);
```

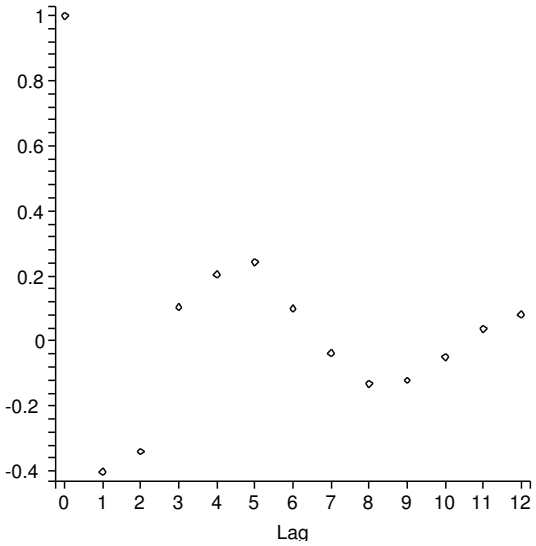
Time series model:

$$\mathtt{Y_t} = \mathtt{c} + \epsilon_\mathtt{t} + \tfrac{1}{5}\epsilon_{\mathtt{t}-1} - \tfrac{7}{5}\epsilon_{\mathtt{t}-2} + \tfrac{11}{5}\epsilon_{\mathtt{t}-3}$$

Autocorrelations:



Partial Autocorrelations:



Unit Roots Analysis:

The roots are:

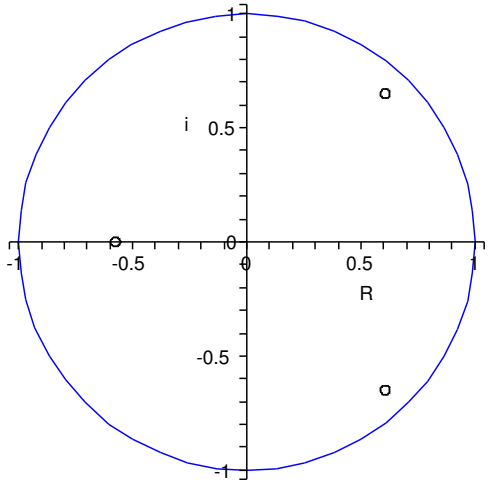
$$\left[-\frac{1}{33}\left(8171+33\sqrt{61305}\right)^{(1/3)} - \frac{16}{33\left(8171+33\sqrt{61305}\right)^{(1/3)}} + \frac{7}{33}, \frac{1}{66}\left(8171+33\sqrt{61305}\right)^{(1/3)} + \frac{8}{33\left(8171+33\sqrt{61305}\right)^{(1/3)}} + \frac{7}{33} \right.$$

$$-\tfrac{1}{2}\mathrm{I}\sqrt{3}\left(-\tfrac{1}{33}(8171+33\sqrt{61305})^{(1/3)}+\tfrac{16}{33(8171+33\sqrt{61305})^{(1/3)}}\right),\ \tfrac{1}{66}(8171+33\sqrt{61305})^{(1/3)}$$
$$+\tfrac{8}{33(8171+33\sqrt{61305})^{(1/3)}}+\tfrac{7}{33}-\tfrac{1}{2}\mathrm{I}\sqrt{3}\left(-\tfrac{1}{33}(8171+33\sqrt{61305})^{(1/3)}+\tfrac{16}{33(8171+33\sqrt{61305})^{(1/3)}}\right)\Bigg]$$

All roots are inside the unit circle.

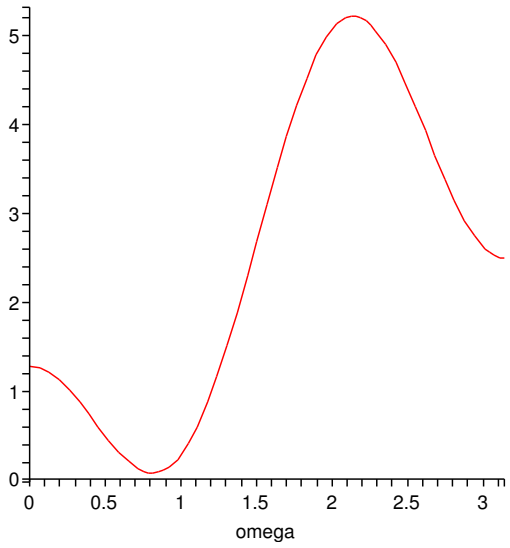
Assuming error terms are i.i.d., then an invertible representation is:

$$\mathrm{Y_t}=\mathrm{c}+\epsilon_{\mathrm{t}}-\frac{7}{11}\epsilon_{\mathrm{t-1}}+\frac{1}{11}\epsilon_{\mathrm{t-2}}+\left(\frac{1}{1089}\sqrt{61305}+\frac{4096}{35937(8171+33\sqrt{61305})}+\frac{8164}{35937}\right)\epsilon_{\mathrm{t-3}}$$

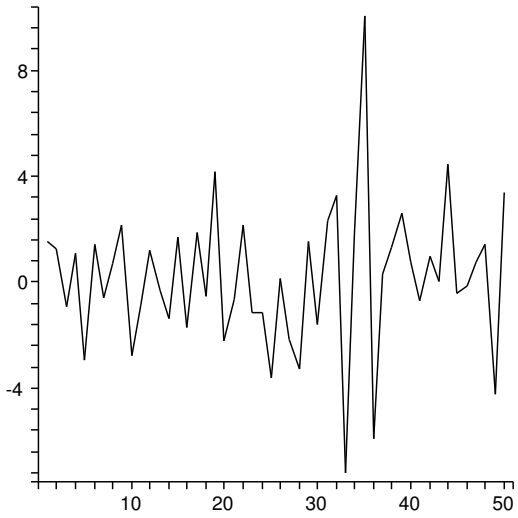


Spectral Density Function:

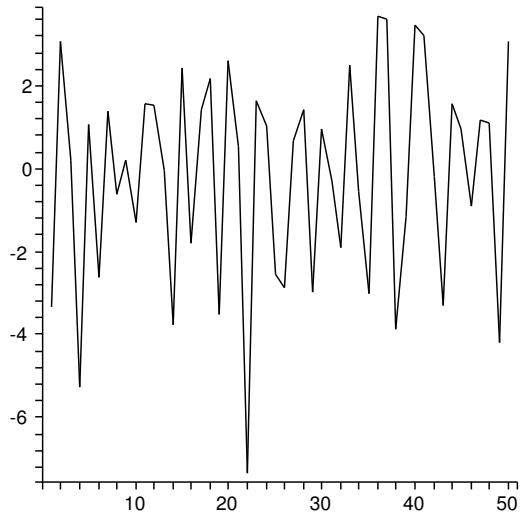
$$\mathrm{f}(\omega)=\frac{\frac{196}{25}-\frac{158}{25}\cos(\omega)-\frac{48}{25}\cos(2\omega)+\frac{22}{5}\cos(3\omega)}{\pi}\qquad 0<\omega<\pi$$



First Realization:



Second Realization:



Acknowledgment

This research is partially supported by an NSF CSUMS grant DMS–0703532 at the College of William & Mary.

References

[1] Box, G., G. Jenkins, *Time Series Analysis: Forecasting & Control (3rd edition).* Prentice Hall, 1994.

[2] Glen, A. G., D. L. Evans, L. M. Leemis, APPL: A Probability Programming Language. *The American Statistician* **55**(2) 156–166. 2001.

[3] Hamilton, J. D., *Time Series Analysis*. Princeton University Press, Princeton, New Jersey, 1994.

[4] Woodward, W. A., H. L. Gray, On The Relationship Between the S Array and the Box–Jenkins Method of ARMA Model Identification. *Journal of the American Statistical Association* **76**(375) 579–587. 1981.