



*The World's Largest Open Access Agricultural & Applied Economics Digital Library*

**This document is discoverable and free to researchers across the globe due to the work of AgEcon Search.**

**Help ensure our sustainability.**

**Give to AgEcon Search**

AgEcon Search

<http://ageconsearch.umn.edu>

[aesearch@umn.edu](mailto:aesearch@umn.edu)

*Papers downloaded from **AgEcon Search** may be used for non-commercial purposes and personal study only. No other use, including posting to another Internet site, is permitted without permission from the copyright owner (not AgEcon Search), or as allowed under the provisions of Fair Use, U.S. Copyright Act, Title 17 U.S.C.*

*No endorsement of AgEcon Search or its fundraising activities by the author(s) of the following work or their employer(s) is intended or implied.*

## From the help desk: Transfer functions

Allen McDowell  
Stata Corporation  
amcdowell@stata.com

**Abstract.** The question often arises as to whether one can estimate a transfer function model using Stata. While Stata does not currently have a convenience command for doing so, this article will demonstrate that estimating such a model can be accomplished quite easily using Stata's `arima` command. The classic text for transfer function modeling is Box, Jenkins, and Reinsel (1994); however, a more concise presentation can be found in Brockwell and Davis (1991).

**Keywords:** `st0009`, `arima`, `xcorr`, `corrgram`, transfer function, impulse-response function, autocorrelation function, cross-correlation function, pre-whitened, linear filter, difference equation

### 1 Transfer function models

Transfer function modeling is simply a methodology for finding a parsimonious, and therefore estimable, parameterization for an infinite-order distributed lag model. In the broadest of terms, transfer functions are a linear filter that can be represented by an equation of the form

$$\begin{aligned} Y_t &= \sum_{i=0}^{\infty} v_i * X_{t-i} + N_t \\ &= \left( \sum_{i=0}^{\infty} v_i * L^i \right) * X_t + N_t \\ &= v(L) * X_t + N_t \end{aligned}$$

whereby deviations of an output from its steady-state level are represented as a linear aggregate of input deviations.  $v(L)$  is a polynomial in the lag operator  $L$  and is known as the transfer function of the filter. The weights,  $v_0, v_1, v_2, \dots$ , are known as the impulse-response function of the system. The term  $N_t$  represents an additive noise term that is assumed to be a random variable that follows a stationary autoregressive-moving average (ARMA) process and is independent of the input  $X_t$ . The system is said to be stable if the  $v_j$  in

$$\sum_{i=0}^{\infty} v_i * L^i$$

are absolutely summable; i.e.,

$$\sum_{i=0}^{\infty} |v_i| < \infty$$

Direct estimation of the  $v$ s is problematic because the right-hand side of the above equation is an infinite series. Even if the  $v$ s are indistinguishable from zero beyond some finite lag length, unless the finite lag structure is known a priori, some method for identifying the lag structure is needed. The method proposed by Box, Jenkins, and Reinsel (1994) is to equate the infinite-order polynomial distributed lag representation of the input-output system to a generalized linear difference equation of the form

$$a(L) * Y_t = c(L) * X_{t-b}$$

so that

$$Y_t = \frac{c(L)}{a(L)} * X_{t-b}$$

Equating coefficients on  $L$  identifies a functional relationship between the  $v$ s of the infinite-order distributed lag representation and the  $a$ s and  $c$ s of the difference equation representation. These functional relationships can become quite complex as the orders of the polynomials  $a(L)$  and  $c(L)$  grow and are not presented here (The interested reader should consult Table 10.1 in Box, Jenkins, and Reinsel (1994). Note that I have also ignored the additive noise term for the moment. Some authors include the noise term throughout the derivation of the model, whereas some simply attach it in an ad hoc fashion. Operationally, it makes no difference which approach one takes; the latter method simplifies exposition without loss of generality.

Perhaps the most confusing aspect of transfer function modeling (or perhaps, better said, the transfer function literature) is the following: while one needs the difference equation representation to develop a strategy for estimating the impulse-response function, one must estimate the impulse-response function before one can estimate the parameters of the difference equation. Again, the problem is that the orders of the polynomials  $a(L)$  and  $c(L)$  and the value of the delay parameter  $b$  are unknown a priori. Since the ultimate goal is to obtain estimates of the impulse-response function, there is no compelling reason to ever estimate the parameters of the difference equation. That is, the role of the difference equation representation is as a conceptual device for obtaining an estimator for the impulse-response function. Once estimates of the impulse-response function are obtained, however, one can obtain estimates of the parameters of the difference equation.

Initial, but inefficient, estimates of the impulse-response function are determined by first prewhitening the input series by applying a linear filter  $d(L)$ . The same filter is then applied to the output series. The cross-correlation function between the two

filtered series will be proportional to the impulse-response function, and the constant of proportion will be the ratio of the standard deviations of the two filtered series. To see that this is true consider the following:

$$Y_t = v(L) * X_t$$

Let  $d(L) * X_t = \alpha_t$  where  $\alpha_t$  is white noise. Applying the same filter,  $d(L)$ , to  $Y_t$  produces

$$d(L) * Y_t = \beta_t$$

where  $\beta_t$  is some stationary ARMA process. Then,

$$d(L) * Y_t = v(L) * d(L) * X_t$$

or equivalently,

$$\beta_t = v(L) * \alpha_t$$

Multiplying both sides by  $\alpha_{t-k}$  and taking expectations yields

$$E(\alpha_{t-k} * \beta_t) = E\{v(L) * \alpha_t * \alpha_{t-k}\}$$

which is

$$\gamma_{\alpha\beta}(k) = v_k * \sigma_\alpha^2$$

where  $\gamma_{\alpha\beta}(k)$  is the cross-covariance at lag +k between  $\alpha_t$  and  $\beta_t$ . Thus,

$$v_k = \frac{\gamma_{\alpha\beta}(k)}{\sigma_\alpha^2}$$

or, in terms of cross-correlations

$$v_k = \frac{\rho_{\alpha\beta}(k) * \sigma_\beta}{\sigma_\alpha} \quad k = 0, 1, 2, \dots$$

Given the initial estimates of the  $v_k$ , one can determine the lag structure of the impulse-response function, including the value of the delay parameter  $b$ . With this information in hand, direct and efficient estimation of the impulse-response function by maximum likelihood is straightforward. This method is demonstrated in the example below.

## 2 Example

The example is taken from Chapter 11 of Box, Jenkins, and Reinsel (1994). A gas furnace is employed in which the input is a variable methane feed rate and the output is the resulting carbon dioxide concentration in the off-gases. In Figure 1 below, we see the time-series plots of the input,  $X_t$ , and the output,  $Y_t$ . Each  $(X_t, Y_t)$  pair was sampled from the continuous records at 9-second intervals.

```
. use furnace, clear
. tsset t
    time variable:  t, 1 to 296
. graph x y t, c(11[...#.#]) s(...) saving(levels, replace)
```

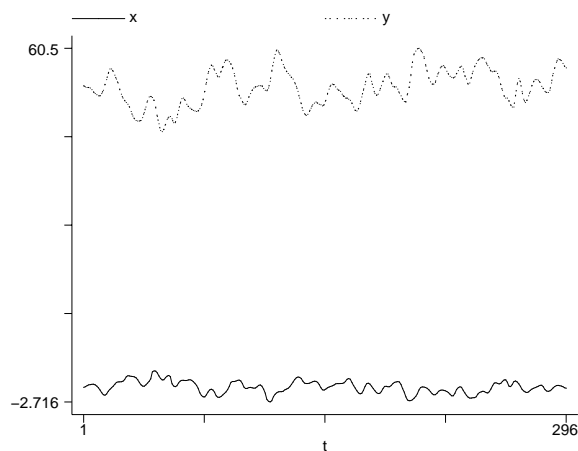


Figure 1: Time-series of  $X_t$  and  $Y_t$

As a preliminary step, we must insure that both series are, in fact, stationary. If the series prove to be nonstationary, then the series must be transformed in order to induce stationarity. We can perform a visual test by examining the correlogram of each series to see if the autocorrelation function decays rapidly. We can also perform statistical tests for the presence of unit roots using either the Dickey–Fuller tests (Stata’s `dfuller` command) or the Phillips–Perron test (Stata’s `pperron` command).

(Continued on next page)

```
. corrgram x, lags(20)
```

LAG	AC	PAC	Q	Prob>Q	-1 [Autocorrelation]	0 [Partial Autocor]
1	0.9525	0.9526	271.26	0.0000		
2	0.8341	-0.7898	480	0.0000		
3	0.6819	0.3424	619.96	0.0000		
4	0.5312	0.1230	705.21	0.0000		
5	0.4075	0.0571	755.55	0.0000		
6	0.3182	-0.1159	786.35	0.0000		
7	0.2602	0.0539	807.01	0.0000		
8	0.2275	0.1030	822.86	0.0000		
9	0.2131	0.0145	836.82	0.0000		
10	0.2083	-0.0712	850.2	0.0000		
11	0.2028	-0.0971	862.94	0.0000		
12	0.1893	0.0455	874.07	0.0000		
13	0.1673	0.0877	882.79	0.0000		
14	0.1375	-0.1436	888.71	0.0000		
15	0.1048	0.0475	892.16	0.0000		
16	0.0754	0.0463	893.95	0.0000		
17	0.0520	-0.0184	894.8	0.0000		
18	0.0371	0.0228	895.24	0.0000		
19	0.0340	0.0944	895.61	0.0000		
20	0.0424	-0.0351	896.18	0.0000		

```
. dfuller x
```

Dickey-Fuller test for unit root

		Interpolated Dickey-Fuller		
		1% Critical Value	5% Critical Value	10% Critical Value
Test Statistic				
Z(t)	-2.665	-3.456	-2.878	-2.570

\* MacKinnon approximate p-value for Z(t) = 0.0802

```
. pperron x
```

Phillips-Perron test for unit root

		Interpolated Dickey-Fuller		
		1% Critical Value	5% Critical Value	10% Critical Value
Test Statistic				
Z(rho)	-33.910	-20.336	-14.000	-11.200
Z(t)	-4.131	-3.456	-2.878	-2.570

\* MacKinnon approximate p-value for Z(t) = 0.0009

While the Phillips-Perron test rejects the null hypothesis of a unit root, the Dickey-Fuller test's rejection is somewhat weak. This is an indication that perhaps we should perform an augmented Dickey-Fuller test just to be sure. The number of lags for the test was chosen to be  $\text{int}(12 * (\frac{T}{100})^{\frac{1}{4}})$ , following Schwert (1989).

```
. dfuller x, lags(2)
Augmented Dickey-Fuller test for unit root      Number of obs   =      293

              Test              Interpolated Dickey-Fuller
Statistic              1% Critical      5% Critical      10% Critical
                        Value              Value              Value
-----
Z(t)              -4.879              -3.457              -2.878              -2.570

* MacKinnon approximate p-value for Z(t) = 0.0000
```

Based on the visual inspection as well as the unit-root tests, we can reject the null hypothesis that there is a unit root in the input series and assume stationarity of the input series  $X_t$ . We can now proceed to test the output series for stationarity.

```
. corrgram y, lags(20)

LAG      AC      PAC      Q      Prob>Q      -1      0      1 -1      0      1
[Autocorrelation] [Partial Autocor]
-----
1      0.9708  0.9747  281.78  0.0000      |-----|
2      0.8960 -0.8575  522.67  0.0000      |-----|
3      0.7925  0.4651  711.77  0.0000      |-----|
4      0.6800  0.2152  851.43  0.0000      |-----|
5      0.5745 -0.0903  951.47  0.0000      |-----|
6      0.4854 -0.0580  1023.2  0.0000      |-----|
7      0.4161 -0.0080  1076  0.0000      |-----|
8      0.3656  0.1319  1116.9  0.0000      |-----|
9      0.3304 -0.0332  1150.5  0.0000      |-----|
10     0.3065 -0.0666  1179.5  0.0000      |-----|
11     0.2880 -0.0937  1205.1  0.0000      |-----|
12     0.2693  0.0666  1227.7  0.0000      |-----|
13     0.2473  0.0236  1246.7  0.0000      |-----|
14     0.2215  0.0200  1262.1  0.0000      |-----|
15     0.1930 -0.0661  1273.7  0.0000      |-----|
16     0.1649  0.0958  1282.3  0.0000      |-----|
17     0.1398  0.0141  1288.5  0.0000      |-----|
18     0.1210  0.0696  1293.1  0.0000      |-----|
19     0.1103 -0.0354  1297  0.0000      |-----|
20     0.1078 -0.0092  1300.7  0.0000      |-----|

. dfuller y
Dickey-Fuller test for unit root      Number of obs   =      295

              Test              Interpolated Dickey-Fuller
Statistic              1% Critical      5% Critical      10% Critical
                        Value              Value              Value
-----
Z(t)              -1.864              -3.456              -2.878              -2.570

* MacKinnon approximate p-value for Z(t) = 0.3492
```

```
. pperron y
Phillips-Perron test for unit root
```

		Number of obs = 295		
		Newey-West lags = 5		
	Test Statistic	1% Critical Value	5% Critical Value	10% Critical Value
Z(rho)	-23.969	-20.336	-14.000	-11.200
Z(t)	-3.423	-3.456	-2.878	-2.570

```
* MacKinnon approximate p-value for Z(t) = 0.0102
```

Note that while the sample autocorrelation function of the output series dampens out fairly quickly and the Phillips–Perron unit root test rejects the null hypothesis of a unit root, the Dickey–Fuller unit root test fails to reject the same null hypothesis. Again, this led me to perform an augmented Dickey–Fuller test using two additional autoregressive lags.

```
. dfuller y, lags(2)
Augmented Dickey-Fuller test for unit root
```

		Number of obs = 293		
	Test Statistic	1% Critical Value	5% Critical Value	10% Critical Value
Z(t)	-3.872	-3.457	-2.878	-2.570

```
* MacKinnon approximate p-value for Z(t) = 0.0023
```

The augmented Dickey–Fuller test rejects the null hypothesis of a unit root so we can proceed, as did Box, Jenkins, and Reinsel (1994), under the assumption that both the input and output series are stationary.

The next step is to pre-whiten the input series. Inspection of the sample autocorrelation function and the sample partial-autocorrelation function suggests an autoregressive model of order 3.

```
. arima x, ar(1/3)
(setting optimization to BHHH)
Iteration 0: log likelihood = 72.554746
Iteration 1: log likelihood = 72.567122
Iteration 2: log likelihood = 72.568408
Iteration 3: log likelihood = 72.568679
Iteration 4: log likelihood = 72.568788
(switching optimization to BFGS)
Iteration 5: log likelihood = 72.568839
Iteration 6: log likelihood = 72.568879
Iteration 7: log likelihood = 72.568895
Iteration 8: log likelihood = 72.568897
Iteration 9: log likelihood = 72.568898
Iteration 10: log likelihood = 72.568898
```



## ARIMA regression

Sample: 1 to 296

Number of obs = 296

Wald chi2(3) = 18662.04

Log likelihood = 72.5689

Prob &gt; chi2 = 0.0000

x	OPG		z	P> z	[95% Conf. Interval]	
	Coef.	Std. Err.				
x _cons	-.0607873	.2112943	-0.29	0.774	-.4749165	.353342
ARMA ar						
L1	1.969063	.0297688	66.15	0.000	1.910717	2.027409
L2	-1.365142	.0690584	-19.77	0.000	-1.500494	-1.22979
L3	.3394078	.0488192	6.95	0.000	.243724	.4350916
/sigma	.1878718	.0040316	46.60	0.000	.17997	.1957736

To get the pre-whitened series, we just use the residuals that are generated by `predict`.

```
. predict alpha, resid
```

We can inspect the correlogram of the filtered series to ensure that we have, in fact, produced a white-noise series.

```
. corrgram alpha, lags(20)
```

LAG	AC	PAC	Q	Prob>Q	-1	0	1	-1	0	1
					[Autocorrelation]			[Partial Autocor]		
1	-0.0352	-0.0352	.36967	0.5432						
2	0.0709	0.0699	1.8759	0.3914						
3	0.0578	0.0632	2.8834	0.4100						
4	-0.1426	-0.1451	9.0243	0.0605						
5	-0.0094	-0.0284	9.051	0.1070						
6	0.0585	0.0792	10.094	0.1208						
7	0.0145	0.0400	10.158	0.1798						
8	0.0024	-0.0286	10.16	0.2540						
9	-0.0542	-0.0766	11.062	0.2715						
10	0.0365	0.0545	11.473	0.3219						
11	0.1434	0.1807	17.841	0.0854						
12	-0.0761	-0.0804	19.64	0.0742						
13	0.0988	0.0320	22.681	0.0457						
14	0.0423	0.0645	23.24	0.0565						
15	-0.0819	-0.0212	25.344	0.0455						
16	0.0172	-0.0304	25.437	0.0625						
17	0.0653	0.0613	26.784	0.0613						
18	-0.0523	-0.0234	27.651	0.0676						
19	-0.0787	-0.1140	29.625	0.0568						
20	0.0229	0.0149	29.792	0.0733						

The evidence does indeed suggest that `alpha` is a white-noise series.

Next, we apply the same filter to the output series. This requires a little data management on our part. This is the type of operation that one would presumably like to see eliminated by a convenience command. Nevertheless, our task is easily accomplished by renaming two variables, using `predict` once again to generate the filtered output series, and then returning our variables back to their original names.

```
. rename x x1
. rename y x
. predict beta, resid
. rename x y
. rename x1 x
```

We will need the standard deviations of the two filtered series for our initial estimates of the impulse-response function, so I will save them as scalars:

```
. summarize alpha
Variable | Obs      Mean   Std. Dev.   Min      Max
-----+-----
alpha    | 296   .0000753   .1882882  -.9202943   .9906531
. scalar s_alpha = r(sd)
. summarize beta in 2/1
Variable | Obs      Mean   Std. Dev.   Min      Max
-----+-----
beta     | 295   3.042023   .377452   2.074999   4.701234
. scalar s_beta=r(sd)
```

We can now examine the cross-correlation function of  $\alpha_t$  and  $\beta_t$ , which, as indicated earlier, will be proportional to the impulse-response function. The `xcorr` command will also allow us to generate a new variable containing the estimates of the cross-correlation function at each lag. From this variable and the standard deviations of  $\alpha_t$  and  $\beta_t$ , we can construct initial estimates of the impulse-response function. From these estimates and an application of Bartlett's approximation of the standard errors of the cross-correlation function (Bartlett 1955), we can identify the lag structure of the transfer function that we wish to estimate.

(Continued on next page)

```
. xcorr alpha beta in 2/1 , saving(alpha_beta, replace) title(" ") ltitle(" ")
```

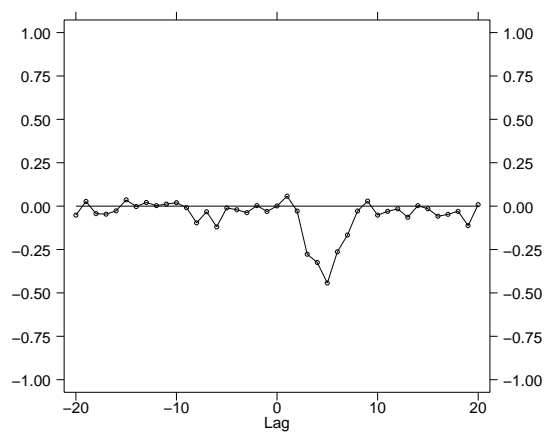


Figure 2: Cross-correlation function of  $\alpha_t$  and  $\beta_t$

```
. xcorr alpha beta in 2/1 , generate(cross) table
```

LAG	CORR	-1	0	1
		[Cross-correlation]		
-20	-0.0519			
-19	0.0269			
-18	-0.0434			
-17	-0.0462			
-16	-0.0272			
-15	0.0358			
-14	-0.0023			
-13	0.0210			
-12	0.0029			
-11	0.0119			
-10	0.0197			
-9	-0.0087			
-8	-0.0965			
-7	-0.0325			
-6	-0.1195			
-5	-0.0103			
-4	-0.0208			
-3	-0.0378			
-2	0.0033			
-1	-0.0306			
0	0.0010			
1	0.0573			
2	-0.0285			
3	-0.2771			
4	-0.3248			
5	-0.4435			
6	-0.2633			
7	-0.1669			

8	-0.0283
9	0.0287
10	-0.0519
11	-0.0305
12	-0.0156
13	-0.0642
14	0.0026
15	-0.0150
16	-0.0583
17	-0.0472
18	-0.0299
19	-0.1122
20	0.0090

We can use the new variable, `cross`, generated by `xcorr` along with the ratio of the standard deviations of the filtered input and output series to get initial estimates of the impulse-response function. Actually, all you need is the estimate of the cross-correlation function to identify the lag structure of the impulse-response function since the impulse-response function is proportional to the cross-correlation function. However, one could use these initial estimates of the impulse-response function as starting values in `arma` for models that have difficulty achieving convergence.

Under the null hypothesis that there is no cross correlation between the pre-whitened input series and the filtered output series, Bartlett's formula for the standard errors of the cross-correlations reduces to  $1/\sqrt{n}$ . While the theoretical impulse-response function will exhibit a pattern dictated by a difference equation of an order equal to 1 plus the order of the polynomial  $a(L)$ , past a certain lag, the individual cross-correlations, and thus the individual impulse-response functions, will not be significantly different from zero. We can, for example, use zero minus two standard deviations as a cutoff to determine which cross-correlations are significantly different from zero. Simulations indicate that with a sample of 296 observations, tests based upon the assumption that the cross-correlations are approximately normally distributed do not have the correct coverage probabilities; the tests are overly conservative. Nevertheless, continuing to follow Box, Jenkins, and Reinsel (1994), we can use the results of these tests to determine the value of  $b$ , the delay parameter, as well as the finite lag structure of the transfer function to be estimated.

```
. generate lag =_n-21
. generate std = -2/sqrt(296)
. generate zero = 0
. generate ir = (s_beta/s_alpha)*cross
(255 missing values generated)
. list lag cross ir std if ir != . & lag >=0 & cross < std
```

	lag	cross	ir	std
24.	3	-.277086	-.5554606	-.1162476
25.	4	-.3247887	-.6510878	-.1162476
26.	5	-.4435291	-.8891206	-.1162476
27.	6	-.2632791	-.5277824	-.1162476
28.	7	-.1669311	-.3346385	-.1162476

From the estimates we can now hypothesize that there is a three-period delay and that the transfer function will include five lags of the input series. We can estimate such a model using `arima` as follows:

```
. arima y l(3/7).x
(setting optimization to BHHH)
Iteration 0:   log likelihood = -363.30663
Iteration 1:   log likelihood = -363.30663
ARIMA regression
Sample:      8 to 296
Log likelihood = -363.3066
Number of obs      =      289
Wald chi2(5)       =    1725.43
Prob > chi2        =      0.0000
```

		Coef.	OPG Std. Err.	z	P> z	[95% Conf. Interval]	
y							
x							
	L3	-.6992201	.3150715	-2.22	0.026	-1.316749	-.0816914
	L4	-.5496065	.7044173	-0.78	0.435	-1.930239	.8310261
	L5	-1.016878	.8318415	-1.22	0.222	-2.647257	.6135019
	L6	.1387479	.6717631	0.21	0.836	-1.177883	1.455379
	L7	-1.014284	.2799281	-3.62	0.000	-1.562933	-.4656348
_cons		53.32959	.0837679	636.64	0.000	53.16541	53.49377
/sigma		.8505925	.0248751	34.19	0.000	.8018382	.8993467

We must now turn our attention to modeling the additive noise term.

```
. predict n, resid
. corrgram n, lags(20)
```

LAG	AC	PAC	Q	Prob>Q	-1	0	1	-1	0	1
						[Autocorrelation]			[Partial Autocor]	
1	0.8850	0.9640	228.7	0.0000						
2	0.6999	-0.5440	372.23	0.0000						
3	0.4924	-0.0406	443.53	0.0000						
4	0.3124	0.1915	472.32	0.0000						
5	0.1818	0.0376	482.11	0.0000						
6	0.0951	-0.0130	484.8	0.0000						
7	0.0385	-0.0852	485.24	0.0000						
8	0.0016	0.0437	485.25	0.0000						
9	-0.0164	0.1025	485.33	0.0000						
10	-0.0085	-0.0079	485.35	0.0000						
11	0.0021	-0.1440	485.35	0.0000						
12	-0.0072	-0.1036	485.37	0.0000						
13	-0.0358	-0.0296	485.76	0.0000						
14	-0.0754	-0.0640	487.49	0.0000						
15	-0.1198	-0.1037	491.9	0.0000						
16	-0.1509	-0.0108	498.91	0.0000						
17	-0.1721	-0.0683	508.07	0.0000						
18	-0.1748	0.0324	517.54	0.0000						
19	-0.1526	0.0541	524.8	0.0000						
20	-0.1021	0.1211	528.06	0.0000						

Inspection of the sample autocorrelation function and the partial autocorrelation function indicates that the noise term follows a second-order autoregressive process. We can now include the error term in our model to jointly estimate all of the parameters.

```
. arima y l(3/7).x, ar(1/2)
(setting optimization to BHHH)
Iteration 0:  log likelihood = -41.695686
Iteration 1:  log likelihood = -23.171625
Iteration 2:  log likelihood = -5.7968769
Iteration 3:  log likelihood = -2.1599672
Iteration 4:  log likelihood = -1.4669091
(switching optimization to BFGS)
Iteration 5:  log likelihood = -1.2421862
Iteration 6:  log likelihood = -.853274
Iteration 7:  log likelihood = -.83456897
Iteration 8:  log likelihood = -.82355348
Iteration 9:  log likelihood = -.82147948
Iteration 10: log likelihood = -.82086731
Iteration 11: log likelihood = -.82067251
Iteration 12: log likelihood = -.82065843
Iteration 13: log likelihood = -.82065789
Iteration 14: log likelihood = -.82065783

ARIMA regression
Sample: 8 to 296                Number of obs   =      289
                                Wald chi2(7)         =    6547.92
Log likelihood = -.8206578      Prob > chi2      =    0.0000
```

		OPG		z	P> z	[95% Conf. Interval]	
y		Coef.	Std. Err.				
y x	L3	-.5557759	.0653626	-8.50	0.000	-.6838843	-.4276675
	L4	-.6444723	.0744984	-8.65	0.000	-.7904865	-.498458
	L5	-.8598032	.0784265	-10.96	0.000	-1.013516	-.7060902
	L6	-.4835769	.0862466	-5.61	0.000	-.6526171	-.3145366
	L7	-.3633083	.0776425	-4.68	0.000	-.5154849	-.2111318
	_cons	53.37647	.1876201	284.49	0.000	53.00874	53.7442
ARMA							
ar	L1	1.537904	.0386701	39.77	0.000	1.462112	1.613696
	L2	-.6291054	.0426486	-14.75	0.000	-.7126951	-.5455156
/sigma		.2413085	.007408	32.57	0.000	.226789	.2558279

For diagnostic purposes, we should now check the residual to ensure that it is a white-noise process and that it is uncorrelated with the filtered input series  $\alpha_t$ . Failure of either of these tests would indicate an inadequacy of the model.

```
. predict residual, residual
(7 missing values generated)
```

```
. corrgram residual, lags(20)
```

LAG	AC	PAC	Q	Prob>Q	-1	0	1	-1	0	1
					[Autocorrelation]			[Partial Autocor]		
1	0.0323	0.0325	.30472	0.5809						
2	0.0710	0.0707	1.7816	0.4103						
3	-0.0749	-0.0801	3.4294	0.3300						
4	-0.0673	-0.0694	4.7668	0.3121						
5	-0.0639	-0.0519	5.9774	0.3084						
6	0.0978	0.1093	8.819	0.1840						
7	0.0272	0.0237	9.04	0.2498						
8	0.0340	0.0065	9.3852	0.3109						
9	-0.0548	-0.0561	10.288	0.3277						
10	0.0893	0.1260	12.692	0.2414						
11	0.0600	0.1142	13.783	0.2453						
12	0.1148	0.1267	17.783	0.1224						
13	-0.0280	-0.0158	18.022	0.1567						
14	0.0634	0.1098	19.252	0.1555						
15	-0.0798	0.0022	21.204	0.1304						
16	-0.0024	0.0232	21.206	0.1707						
17	-0.0745	-0.0738	22.924	0.1517						
18	-0.0112	-0.0585	22.963	0.1920						
19	-0.1106	-0.1399	26.772	0.1101						
20	-0.0003	-0.0691	26.772	0.1418						

```
. xcorr alpha residual, saving(alpha_residual, replace) title(" ") l1title(" ")
```

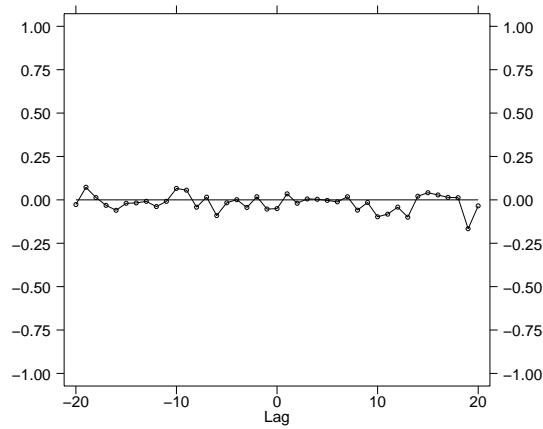


Figure 3: Cross-correlation function of residual and  $\alpha_t$

The diagnostics indicate that the residual is white noise and that it is uncorrelated with the input series  $\alpha_t$ .

Finally, we can generate the predicted value of the output and compare it to the observed series.

```
. predict yhat
(option xb assumed; predicted values)
(7 missing values generated)
. graph y yhat t, c(11[...#]) s(...) saving(prediction, replace)
```

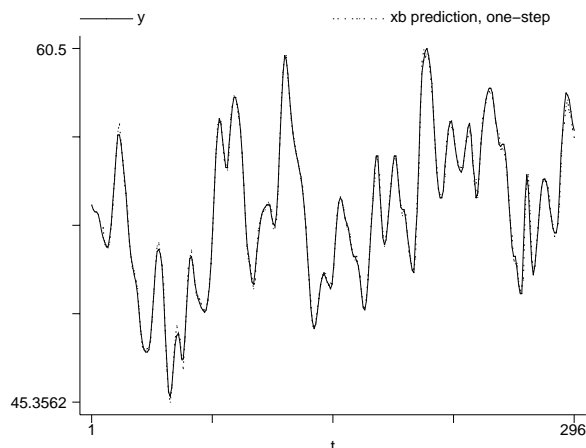


Figure 4: Time-series of Output (Y) and Prediction (yhat)

Comparison of the parameter estimates achieved by `arima` and those reported by Box, Jenkins, and Reinsel (1994) indicates that we have estimated the same model. There are minor numerical differences, but these differences are due to rounding in the reported results in Box, Jenkins, and Reinsel (1994).

### 3 References

- Bartlett, M. S. 1955. *Stochastic Processes*. Cambridge: Cambridge University Press.
- Box, G. E., G. M. Jenkins, and G. C. Reinsel. 1994. *Time Series Analysis – Forecasting and Control*. 3d ed. Upper Saddle River, NJ: Prentice-Hall.
- Brockwell, P. J. and R. A. Davis. 1991. *Time Series: Theory and Methods*. 2d ed. New York: Springer-Verlag.
- Schwert, W. G. 1989. Tests for Unit Roots: A Monte Carlo Investigation. *Journal of Business & Economic Statistics* 7(2).

#### About the Author

Allen McDowell is Director of Technical Services at Stata Corporation.