Implementing Maximum Likelihood for Models with Autoregressive Disturbances

Russell Davidson

Table of Contents

Table of Contents **iii**

1 Maximum Likelihood with AR Disturbances **1**

1 Introduction 1

2 The initial observations 3

3 The Artificial Regression 7

4 An Illustrative Example 15

Code Index **17**

Chapter 1

Maximum Likelihood with AR Disturbances

1. Introduction

Recall the artificial regression that corresponds to the maximum likelihood estimator of the model

$$\boldsymbol{y} = \boldsymbol{x}(\boldsymbol{\theta}) + \boldsymbol{u}; \quad \boldsymbol{u} \sim \mathrm{N}(\mathbf{0}, \boldsymbol{\Omega}(\theta)). \tag{1}$$

In order to formulate the artificial regression, we construct the $n \times n$ lower-triangular matrix $\boldsymbol{A}(\boldsymbol{\theta})$, such that $\boldsymbol{A}^\top(\boldsymbol{\theta})\boldsymbol{A}(\boldsymbol{\theta}) = \boldsymbol{\Omega}^{-1}(\boldsymbol{\theta})$. The residual vector $\boldsymbol{u}(\boldsymbol{\theta})$ is defined as usual as $\boldsymbol{u}(\boldsymbol{\theta}) = \boldsymbol{y} - \boldsymbol{x}(\boldsymbol{\theta})$, and the vector of homoskedastic, serially uncorrelated, elementary zero functions for the first moment of $\boldsymbol{y}$ is just $\boldsymbol{v}(\boldsymbol{\theta}) \equiv \boldsymbol{A}(\boldsymbol{\theta})\boldsymbol{u}(\boldsymbol{\theta})$. The second-moment zero functions, also homoskedastic, with the same variance as the elements of $\boldsymbol{v}$, also serially uncorrelated, and uncorrelated with the first-moment functions, are the functions $(v_t^2(\boldsymbol{\theta}) - 1)/\sqrt{2}$, $t = 1, \ldots, n$.

Each observation contributes two pseudo-observations to the regression, one for the first moment, the other for the second. For observation t, these pseudo-observations look like:

$$\begin{bmatrix} v_t \\ (v_t^2 - 1)/\sqrt{2} \end{bmatrix} = \sum_{i=1}^{k} \begin{bmatrix} R_{ti}^{(1)} \\ R_{ti}^{(2)} \end{bmatrix} b_i + \text{residuals}, \tag{2}$$

where

$$R_{ti}^{(1)} = -\sum_{s=1}^{t} a_{ts}\frac{\partial u_s}{\partial \theta_i} - \sum_{s=1}^{t} u_s \frac{\partial a_{ts}}{\partial \theta_i} + \frac{1}{a_{tt}}\frac{\partial a_{tt}}{\partial \theta_i} v_t, \text{ and} \tag{3}$$

$$R_{ti}^{(2)} = -\sqrt{2}\,\frac{1}{a_{tt}}\frac{\partial a_{tt}}{\partial \theta_i}. \tag{4}$$

I propose to implement this artificial regression, in a somewhat simplified form, for the linear regression model with AR(p) disturbances:

$$\boldsymbol{y} = \boldsymbol{X\beta} + \boldsymbol{u}, \text{ where}$$
$$u_t = \sum_{i=1}^{p} \rho_i u_{t-i} + v_t, \tag{5}$$

the v_t being supposed to constitute a Gaussian white noise process. In addition, we suppose that the disturbance process u_t is stationary, and not just asymptotically stationary, so that we know the joint distribution of the first p elements, $u_1, \ldots, u_p$, which are not fully specified by the recursion (5).

First consider the observations $p+1, \ldots, n$. Since the v_t are white noise, rows $p+1$ to n of $\boldsymbol{A}$ follow directly from (5). We have

$$a_{ts} = \delta_{ts} - \sum_{i=1}^{p} \rho_i \delta_{s,t-i}. \tag{6}$$

Here, I implicitly assume that the variance of v_t is 1. It is easy enough to take account later of a non-unit variance that has to be estimated. The parameters fall into two groups: the elements of the k–vector $\boldsymbol{\beta}$, and the elements of the p–vector $\boldsymbol{\rho}$, with elements ρ_i, $i = 1, \ldots, p$. The residuals u_t depend only on $\boldsymbol{\beta}$; the elements a_{ts} only on $\boldsymbol{\rho}$. For an element of β, therefore, the regressor (4) is zero. Now $\partial u_s / \partial \beta_i = -X_{si}$, and so the regressor (3) for β_i is

$$\sum_{s=1}^{t} a_{ts} X_{si} = X_{ti} - \sum_{j=1}^{p} \rho_j X_{t-j,i}. \tag{7}$$

For ρ_j, the second-moment regressor is zero, because $a_{tt} = 1$, and its derivative is 0. From (6), we see that the first-moment regressor is

$$-\sum_{s=1}^{t} u_s \frac{\partial a_{ts}}{\partial \rho_j} = u_{t-j} \tag{8}$$

So far, then, this is just what we would do with a plain GNR.

Regarding estimating the variance, we can redefine the v_t as the old value divided by σ. The instrument for σ for the first-moment zero functions is then 0. The second-moment functions become $(v_t^2 - \sigma^2)/\sigma\sqrt{2}$, (old definition of v_t), and so the instrument for σ^2 for them is just a constant independent of t.

2. The initial observations

We now have to construct elementary zero functions for the first p observations. Doing this requires knowledge of the joint distribution of the disturbances $u_1, \ldots, u_p$. Under the assumption of normality, this means knowledge just of the covariance matrix of these variables. Under the assumption of stationarity, this covariance matrix is a Toeplitz matrix defined using the stationary variance and covariances of an AR(p) process. The quantities can be obtained from the Yule-Walker equations. Denoting the variance by v_0 and the covariance at distance j by v_j, we can write these equations for an AR(p) process as

$$\begin{aligned} v_0 - \sum_{j=1}^{p} \rho_j v_j &= 1, \quad \text{and} \\ v_i - \sum_{j=1}^{p} \rho_j v_{|i-j|} &= 0, \quad i = 1, \ldots, p. \end{aligned} \tag{9}$$

We have $p+1$ equations for the $p+1$ unknowns $v_0, v_1, \ldots v_p$. Note that the left-hand side of the first equation (we index it by 0, not 1) takes exactly the same form as that of the second equation, setting $i = 0$.

Despite the compact form of (9), it is not very convenient to program the set of equations in the form of a matrix of coefficients times the vector of the v_j equal to a right-hand side vector. The difficulty arises because, in equation i of (9), it is the coefficients ρ_j that are indexed directly, while the unknowns are, except for v_i itself, indexed indirectly with index $|i-j|$.

Consider first the case with $i \geq j$. Then $|i-j| = i-j$. Numbering rows and columns from 0, the coefficient ρ_j of the term $\rho_j v_{|i-j|}$ belongs to the row (equation) indexed by i, and the column (variable) indexed by $i-j$. Thus j is the row index minus the column index. Next, let $i < j$. Then $|i-j| = j-i$ is the column index. It follows in this case that j is equal to the row index plus the column index.

For convenience, we may define a $p+1$–vector $\boldsymbol{\rho}$ with zeroth element equal to 0. The sums over j in (9) can thus be extended so as to run from 0 to p. Then, for $r, c = 0, 1, \ldots, p$, element (r, c) of the coefficient matrix of the Yule-Walker equations (9) is

$$\delta_{rc} - \rho(\max(0, r-c)) - (1 - \delta_{0c})\rho(\max(0, r+c)), \tag{10}$$

where $\rho(i)$ denotes element i of $\boldsymbol{\rho}$, that is, ρ_i for $i > 0$ and 0 for $i = 0$. Observe that if $r > c > 1$ both of the last two terms above are nonzero. This corresponds to the fact that, for given i, $|i-j|$ is the same positive integer for $j = i+k$ and $j = i-k$. The factor of $1 - \delta_{0c}$ prevents double counting of the term $\rho(r)$ when $c = 0$. Indeed, there is only one term in v_0 in any of the equations.

Since in ***Ects*** row and column indexing starts at 1 rather than 0, the implementation of (10) is as follows.

⟨*Yule-Walker coefficient matrix*⟩≡

```
mat YW(Row, Column) = (Row=Column)-\
   rho(1+max(0,Row-Column))-(Column>1)*rho(1+max(0,Row+Column-2))
```

The matrix `YW` is previously set up as a $(p+1)\times(p+1)$ matrix.

⟨*Create YW*⟩≡

```
mat YW = emptymatrix(p+1,p+1)
```

Solving equations (9) is now straightforward. We formulate the right-hand side vector with 1 in first position, 0 elsewhere, and get a vector $\boldsymbol{V}$ of variances and covariances by inverting the `YW` matrix.

⟨*Solve Yule-Walker equations*⟩≡

```
sample 1 p+1
gen rhs = 0
set rhs(1) = 1
mat YWinv = YW inv
mat V = YWinv*rhs
```

The variances and covariances can now be put into matrix form, and the Cholesky decomposition of $\boldsymbol{\Omega}^{-1}$ computed.

⟨*Get matrix A*⟩≡

```
mat Om(Row,Column) = V(abs(Row-Column)+1)
mat A = uptriang(Om inv)'
```

Here $\boldsymbol{\Omega}$ (that is, `Om`) is initialised as a $p\times p$ matrix.

⟨*Initialisation of Om*⟩≡

```
mat Om = emptymatrix(p,p)
```

The derivatives of $\boldsymbol{A}$

The easiest way to obtain the derivatives of the elements a_{ts} of the matrix $\boldsymbol{A}$ with respect to the parameters $\boldsymbol{\beta}$ and $\boldsymbol{\rho}$, needed for the regressors (3) and (4), is first to obtain the derivatives of $\boldsymbol{\Omega}$. This can be done by differentiating the Yule-Walker equations. If we differentiate equation i, $i=0,1,\ldots,p$, of (9) with respect to ρ_k, $k=1,\ldots,p$, we get

$$\frac{\partial v_i}{\partial\rho_k}-\sum_{j=1}^{p}\rho_j\frac{\partial v_{|i-j|}}{\partial\rho_k}=v_{|i-k|}.$$

Considered as a set of $p+1$ equations for the $p+1$ unknowns $\partial v_i/\partial\rho_k$, $i=0,1,\ldots,p$, k fixed, the coefficient matrix is identical to that used in the Yule-Walker equations (9) themselves. Thus all we need do is to set up a right-hand side vector with element $v_{|i-k|}$ in position i. For $k=1$, for instance, this vector is $[v_1 \vdots v_0 \vdots v_1 \vdots \ldots \vdots v_{p-1}]$. For $k=p$ it is $[v_p \vdots v_{p-1} \vdots \ldots \vdots v_0]$. Remembering that ***Ects*** indices begin at 1 not 0, we see that the right-hand side for the derivatives with respect to ρ_k is obtained by the following command, because `V` is the vector of the v_i, $i=0,1,\ldots,p$.

$\langle$*rhs for derivatives of Omega*$\rangle\equiv$

```
mat rhs = rowcat(reverse(V(2,k+1,1,1)),V(1,p+1-k,1,1))
```

The vector of derivatives with respect to ρ_k is obtained by solving the equations with this right-hand side vector. The derivative of the matrix $\boldsymbol{\Omega}$ is obtained by arranging these derivative just as the elements of `V` were arranged into $\boldsymbol{\Omega}$ itself.

$\langle$*derivative matrix of Omega*$\rangle\equiv$

```
mat d = YWinv*rhs
mat dV(Row,Column) = d(abs(Row-Column)+1)
```

The matrix `dV` is previously constructed as a $p\times p$ matrix.

$\langle$*Create dV*$\rangle\equiv$

```
mat dV = emptymatrix(p,p)
```

Since $\boldsymbol{A}^\top\boldsymbol{A}=\boldsymbol{\Omega}^{-1}$, it follows that

$$\frac{\partial\boldsymbol{A}^\top}{\partial\rho_k}\boldsymbol{A}+\boldsymbol{A}^\top\frac{\partial\boldsymbol{A}}{\partial\rho_k}=\frac{\partial\boldsymbol{\Omega}^{-1}}{\partial\rho_k}=-\boldsymbol{\Omega}^{-1}\frac{\partial\boldsymbol{\Omega}}{\partial\rho_k}\boldsymbol{\Omega}^{-1}=-\boldsymbol{A}^\top\boldsymbol{A}\frac{\partial\boldsymbol{\Omega}}{\partial\rho_k}\boldsymbol{A}^\top\boldsymbol{A}.$$

Premultiplying by $(\boldsymbol{A}^\top)^{-1}$ and postmultiplying by $\boldsymbol{A}^{-1}$ gives

$$(\boldsymbol{A}^\top)^{-1}\frac{\partial\boldsymbol{A}^\top}{\partial\rho_k}+\frac{\partial\boldsymbol{A}}{\partial\rho_k}\boldsymbol{A}^{-1}=-\boldsymbol{A}\frac{\partial\boldsymbol{\Omega}}{\partial\rho_k}\boldsymbol{A}^\top.$$

Since $\boldsymbol{A}$ is lower-triangular, the first term on the left-hand side here is upper-triangular, being the transpose of the second, lower-triangular term. The right-hand side is, of course, symmetric. It is computed by the command

$\langle$*RHS of derivative equation*$\rangle\equiv$

```
mat RHS = -A*dV*A'
```

We want to treat the diagonal and off-diagonal parts of `RHS` separately. The diagonal part is extracted, as a diagonal matrix, by the command

⟨*diagonal of RHS*⟩≡

```
mat dg = makediag(diag(RHS))
```

The function `diag` creates a vector containing the diagonal elements of its argument; the function `makediag` converts a vector into a diagonal matrix.

The lower triangle of `RHS`, without the principal diagonal, is the corresponding part of $(\partial \boldsymbol{A}/\partial \rho_k)\boldsymbol{A}^{-1}$, whereas the diagonal part of $(\partial \boldsymbol{A}/\partial \rho_k)\boldsymbol{A}^{-1}$ is half the diagonal of `RHS`. Thus we can get the lower-triangular matrix $\partial \boldsymbol{A}/\partial \rho_k$ by the following manipulations.

⟨*Compute dA*⟩≡

```
mat RHS(Row,Column) = (Row>Column)*RHS(Row,Column)
mat dA = (RHS+dg/2)*A
```

We choose to return $\boldsymbol{A}$ and the full set of its derivatives $\partial \boldsymbol{A}/\partial \rho_k$, $k = 1, \ldots, p$, in one $p \times p(p+1)$ matrix constructed as follows:

$$\begin{bmatrix} \boldsymbol{A} & \dfrac{\partial \boldsymbol{A}}{\partial \rho_1} & \cdots & \dfrac{\partial \boldsymbol{A}}{\partial \rho_p} \end{bmatrix}.$$

To this end, we initialise a matrix `AdA` by $\boldsymbol{A}$.

⟨*initialise AdA*⟩≡

```
mat AdA = A
```

Then we loop over k, appending the derivative of $\boldsymbol{A}$ with respect to ρ_k on each iteration.

⟨*Compute AdA*⟩≡

```
set k = 0
while k < p
   set k = k+1
   ⟨rhs for derivatives of Omega⟩
   ⟨derivative matrix of Omega⟩
   ⟨RHS of derivative equation⟩
   ⟨diagonal of RHS⟩
   ⟨Compute dA⟩
   mat AdA = colcat(AdA,dA)
end
```

The function `ARp`

A function is now defined, called `ARp`, which returns the $p \times p(p+1)$ matrix just constructed. It determines the lag order p for the passed argument, and then inserts a first element of 0 for the calculations that follow.

⟨*ARp function*⟩≡

```
function AdA ARp(rrho)
   local p, rho, YW, rhs, V, Om, A, i, d, dV, RHS, dg, dA, k
   set p = rows(rrho)
   mat rho = rowcat(0,rrho)
   ⟨Create YW⟩
   ⟨Initialisation of Om⟩
   ⟨Create dV⟩
   ⟨Yule-Walker coefficient matrix⟩
   ⟨Solve Yule-Walker equations⟩
   ⟨Get matrix A⟩
   ⟨initialise AdA⟩
   ⟨Compute AdA⟩
end
```

3. The Artificial Regression

We now have all the necessary ingredients for the artificial regression characterised by (2). We suppose that the dependent variable for the model (1) is available as `y`, and we suppose further (for simplicity of the programming) that the regression function vector $\boldsymbol{x}(\boldsymbol{\beta})$ is linear, of the form $\boldsymbol{X}\boldsymbol{\beta}$, where the $n \times k$ matrix $\boldsymbol{X}$ is in memory under the name `X`. The dimensions of this matrix are obtained and given the usual names of `n`, for the sample size, and `k` for the number of parameters.

⟨*Get dimensions of problem*⟩≡

```
set n = rows(X)
set k = cols(X)
```

The artificial regression requires that the parameters should be initialised, in order that the artificial variables may be evaluated. The starting point for the regression parameters is obtained by simply running an OLS regression of $\boldsymbol{y}$ on $\boldsymbol{X}$.

⟨*initialise beta*⟩≡

```
sample 1 n
ols y X
mat beta = coef
```

For future use, we save the residuals from this regression (as `u`) and the standard error of the regression (as `s`).

⟨*Residuals and standard error*⟩≡

```
gen u = res
set s2 = errvar
set s = sqrt(s2)
```

Before we can embark on initialising the $p-$vector $\boldsymbol{\rho}$, we must suppose that the value of p has been supplied as the variable `p`. Then we run the regression with typical observation

$$\hat{u}_t = \sum_{i=1}^{p} \rho_i \hat{u}_{t-i} + \text{residual} \tag{11}$$

for observations $p+1$ through n. The regressors are put into a matrix `U`, which has a full n rows, so as not to waste potentially useful information.

⟨*Matrix of lagged residuals*⟩≡

```
mat U = emptymatrix()
set i = 0
while i < p
   set i = i+1
   gen U = colcat(U,lag(i,u))
end
```

Thus the regression:

⟨*initialise rho*⟩≡

```
sample p+1 n
ols u U
mat rho = coef
```

The homoskedastic, serially uncorralated, elementary zero functions are the elements of the vector $\boldsymbol{v} \equiv \boldsymbol{A}\boldsymbol{u}$. Here $\boldsymbol{u}$ is just the vector of residuals $\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}$. Given the expression (6) for the elements a_{ts} for $t > p$, we see that, for $t > p$,

$$v_t = u_t - \sum_{i=1}^{p} \rho_i u_{t-1}, \tag{12}$$

so that, for the initial values of $\boldsymbol{\beta}$ and ρ, all but the first p elements of $\boldsymbol{v}$, which constitute a subvector of the regressand of the artificial regression, are given by `u` minus the fitted values from regression (11). We compute this vector, and select rows $p+1$ through n of the matrix `U` of lagged residuals, since, from (8), they are the regressors corresponding to the components of $\boldsymbol{\rho}$.

⟨*Parts of the artificial regression*⟩≡

```
gen r = u-fit
mat r = r(p+1,n,1,1)
mat U = U(p+1,n,1,p)
mat X1 = X(1,p,1,k)
```

As we see, it is also useful to extract from the matrix $\boldsymbol{X}$ the rows that correspond to the inital observations.

The entire intialisation phase can now be assembled.

⟨*initialisation phase*⟩≡

⟨*Get dimensions of problem*⟩
⟨*initialise beta*⟩
⟨*Residuals and standard error*⟩
⟨*Matrix of lagged residuals*⟩
⟨*initialise rho*⟩
⟨*Parts of the artificial regression*⟩

The iterative loop

Still for observations $p + 1$ through n, we can compute the regressors (7) that correspond to $\boldsymbol{\beta}$. We use a loop for this purpose. Again, we perform the operation for the entire sample, so as not to lose needed information inadvertently.

⟨*Regressors for beta*⟩≡

```
sample 1 n
gen Rb = X
set i = 0
while i < p
   set i = i+1
   gen Rb = Rb-lag(i,X)*rho(i)
end
```

Now we must set about computing regressors and regressand for obervations 1 through p, each of which contributes two observations to the artificial regression. We start by calling the function `ARp` for the current vector $\boldsymbol{\rho}$.

⟨*first p observations*⟩≡

```
sample 1 p
mat ada = ARp(rho)
```

We recall that the first $p \times p$ block of the returned matrix is the matrix we call $\boldsymbol{A}$. For the second-moment regressors (4), we need the diagonal elements of this matrix.

⟨first p observations⟩+≡

```
mat A = ada(1,p,1,p)
mat at = diag(A)
```

In order to make use of the derivatives of $\boldsymbol{A}$ returned by the function `ARp`, we set up another loop in which we construct, first, a $p \times p$ matrix `dA`, column i of which is the vector of derivatives with typical element $\partial a_{tt}/\partial \rho_i$, and, second, a $p \times p$ matrix `Rr1`: it will be the matrix of regressors (`R`) corresponding to $\boldsymbol{\rho}$ (`r`) for the first-moment (`1`) zero functions. After the current loop is finished, element ti of `Rr1` is $-\sum_{s=1}^{t} u_s \partial a_{ts}/\partial \rho_i$.

⟨first p observations⟩+≡

```
mat dA = emptymatrix()
mat Rr1 = emptymatrix()
gen up = u
set i = 0
while i < p
   set i = i+1
   mat dA = colcat(dA,diag(ada(1,p,p*i+1,p*(i+1))))
   mat Rr1 = colcat(Rr1,-ada(1,p,p*i+1,p*(i+1))*up)
end
```

Note that `up` is just the first p elements of the residual vector `u`.

Next, we generate the $p \times p$ matrix `dlogA` with typical element $(1/a_{tt})\partial a_{tt}/\partial \rho_i$. We use a `gen` command for this, as it just divides each column of `dA` element by element by the corresponding element of the vector `at`.

⟨first p observations⟩+≡

```
gen dlogA = dA/at
```

From (3), we can see that the first-moment regressor for ρ_i is

$$-\sum_{s=1}^{t} u_s \frac{\partial a_{ts}}{\partial \rho_i} + \frac{1}{a_{tt}} \frac{\partial a_{tt}}{\partial \rho_i} v_t. \tag{13}$$

Now `Rr1` has been computed as the matrix of which element ti is the first term above. For the second term, we need the vector $\boldsymbol{v}$, which is also of course the first-moment regressand. For observations 1 through p, we compute it (as `r1`) by premultiplying `up` by `A`.

⟨first p observations⟩+≡

```
mat r1 = A*up
```

Then we can compute the second term of (13) and add it to `Rr1`.

⟨*first p observations*⟩+≡

```
gen rr1 = dlogA*r1
mat Rr1 = Rr1+rr1
```

For $\boldsymbol{\beta}$, we see from (3) that the first-moment regressor for β_i is

$$-\sum_{s=1}^{t} a_{ts}\frac{\partial u_s}{\partial \beta_i} = \sum_{s=1}^{t} a_{ts}X_{si},$$

and so we compute `Rb1` as follows:

⟨*first p observations*⟩+≡

```
mat Rb1 = A*X1
```

Each element of $\boldsymbol{v}$ has variance σ^2. We require therefore that the second-moment zero functions should be scaled so as to have the same variance. The numerator of a typical second-moment zero function is $v_t^2 - \sigma^2$, of which the variance, under the supposed normality of v_t, is $2\sigma^4$. If we divide the numerator by $\sigma\sqrt{2}$, the variance of the ratio becomes σ^2, as required. The second-moment regressand, denoted `r2`, is thus given by

⟨*first p observations*⟩+≡

```
gen r2 = (r1^2-s^2)/(s*rt2)
```

The second-moment regressors for $\boldsymbol{\rho}$ then follow from (4), slightly modified to take account of the variance. Since the second-moment zero function is $(v_t^2 - \sigma^2)/\sigma\sqrt{2}$, the optimal moment for ρ_i is the negative of the expectation of the derivative of this function with respect to ρ_i. We have

$$\frac{\partial}{\partial \rho_i}\frac{v_t^2 - \sigma^2}{\sigma\sqrt{2}} = \frac{\sqrt{2}}{\sigma}v_t\frac{\partial v_t}{\partial \rho_i}.$$

Now

$$v_t\frac{\partial v_t}{\partial \rho_i} = \sum_{s=1}^{t}\frac{\partial a_{ts}}{\partial \rho_i}u_s v_t,$$

and $\mathrm{E}(u_s v_t) = \delta_{ts}/a_{tt}$ conditional on information at time t. It follows that the optimal instrument is

$$-\mathrm{E}\left(\frac{\partial}{\partial \rho_i}\frac{v_t^2 - \sigma^2}{\sigma\sqrt{2}}\right) = -\frac{\sqrt{2}}{\sigma a_{tt}}\frac{\partial a_{tt}}{\partial \rho_i},$$

which is just (4) divided by σ. It is evaluated as follows.

⟨*first p observations*⟩+≡

```
mat Rr2 = -dlogA*rt2/s
```

We recall that the corresponding regressors for $\boldsymbol{\beta}$ are zero.

Because it is used repeatedly, the quantity `rt2` is computed once for all.

⟨*rt2 definition*⟩≡

```
set rt2 = sqrt(2)
```

We are now ready to assemble all three parts of the variables of the artificial regression. We have, on top, the contributions from observations $p+1$ through n, from first-moment zero functions only, since, as we recall, the second-moment functions are informative only about σ^2. Below we have p contributions from the first-moment functions for the first p observations, and below that p contributions from the second-moment functions for these same observations. The second-moment functions are uninformative about $\boldsymbol{\beta}$, and so there is no `Rb2` needed.

⟨*the artificial regression*⟩≡

```
mat Rb = rowcat(Rb(p+1,n,1,k),Rb1)
mat Rr = rowcat(U,Rr1,Rr2)
mat r = rowcat(r,r1,r2)
```

The artificial regression is now run, and the estimates of the artificial parameters used to update $\boldsymbol{\beta}$ and $\boldsymbol{\rho}$.

⟨*the artificial regression*⟩+≡

```
sample 1 n+p
ols r Rb Rr
gen rsave = r
mat beta = beta+coef(1,k,1,1)
mat rho = rho+coef(k+1,k+p,1,1)
```

The regressand is saved in `rsave` because, once convergence is achieved, we want to rerun the artificial regression in order to display the results.

We can use the fit of the artificial regression as a measure of how near we are to convergence. The square root of the explained sum of squares is the actual number we use. Since there is a small but positive risk that the esplained sum of squares might evaluate to minus zero, we take the absolute value so as to avoid floating point problems.

⟨*the artificial regression*⟩+≡

```
sets crit = sqrt(abs(sse))
```

Note the use of the command `sets` rather than plain `set`. This causes the value of `crit` to be printed in the log, so that the user can check whether convergence is being achieved.

We recall that, when we entered the iterative loop, the residuals `u`, along with the matrix `U` of their first p lags, had already been computed for observations $p+1$ through n. Inaddition, these had been used to compute the regressand `r` for these observations. We want, therefore, to update these quantities before moving on to the next iteration. For the regressand, we use formula (12) to perform the update.

⟨*update residuals and main part of regressand*⟩≡

```
sample 1 n
mat u = y-X*beta
mat U = emptymatrix()
set i = 0
while i < p
   set i = i+1
   gen U = colcat(U,lag(i,u))
end
mat U = U(p+1,n,1,p)
mat r = u(p+1,n,1,1)-U*rho
```

The other parameter that was already evaluated on entry into the loop is σ, and so we must also update it. The first-moment functions are never informative about this parameter. The derivative of the second-moment function $(v_t^2 - \sigma^2)/\sigma\sqrt{2}$ with respect to σ is (recall that, with our definition, v_t does not depend on σ)

$$\frac{\partial}{\partial\sigma}\frac{v_t^2-\sigma^2}{\sigma\sqrt{2}} = -\frac{v_t^2+\sigma^2}{\sigma^2\sqrt{2}},$$

and so the optimal instrument, which is the negative of the expectation of this, is just $\sqrt{2}$. Thus the estimating equation for σ can be written as

$$\frac{1}{\sigma}\sum_{t=1}^{n}(v_t^2-\sigma^2) = 0,$$

with solution $\hat\sigma^2 = \frac{1}{n}\sum_{t=1}^{n} v_t^2$.

Strictly speaking, we should also update the first-moment zero functions for observations 1 through p. However, it is convenient to use their values from the current iteration before updating, since recomputing them involves a call to the function `ARp`. At convergence, updating the zero functions does not change them, and so this time-saving trick does not matter. The update of σ^2 is thus as follows.

⟨update sigma⟩≡

```
mat s2 = (r'*r+r1'*r1)/n
set s = sqrt(s2)
```

The entire iterative loop can now be set up.

⟨iterative loop⟩≡

```
set iter = 0
set crit = 1
while (iter<maxiter)*(crit>tol)
   set iter = iter+1
   ⟨Regressors for beta⟩
   ⟨first p observations⟩
   ⟨the artificial regression⟩
   ⟨update residuals and main part of regressand⟩
   ⟨update sigma⟩
end
```

The scalar variable `maxiter` should be initialised before beginning the loop so that, if convergence cannot be achieved, we do not sit waiting for infinite time. The convergence criterion `tol` should also be set as desired.

Completing the program

The loop is run with output turned off, so as to avoid excessive and uninformative material in the output file and the log. Once convergence is achieved, we want to see what the results are, and so we rerun the last iteration of the artificial regression in order to get standard errors and the estimated covariance matrix, and just to check that convergence has indeed been attained. The final parameter estimates are then printed, and the number of iterations displayed in the log.

⟨final output⟩≡

```
sample 1 n+p
ols rsave Rb Rr
show iter
sample 1 k
print beta
sample 1 p
print rho
print s2
```

We can now put together a file, `arml.ect`, which, if included after reading in variables and setting the values of the lag length p, `maxiter`, and `tol`, will perform the Gaussian maximum likelihood estimation of the AR(p) model.

$\langle$*arml.ect*$\rangle\equiv$

```
silent
noecho
```

$\langle$*ARp function*$\rangle$

$\langle$*rt2 definition*$\rangle$
$\langle$*initialisation phase*$\rangle$
$\langle$*iterative loop*$\rangle$

```
restore
```
$\langle$*final output*$\rangle$

4. An Illustrative Example

The file `ar3.dat` contains data generated by the following ARX(3) process:

$$\begin{aligned} y_t &= \boldsymbol{X}_t\boldsymbol{\beta} + u_t, \\ u_t &= 0.1u_{t-1} + 0.2u_{t-2} + 0.3u_{t-3} + 20v_t, \qquad v_t \sim \text{NID}(0,1). \end{aligned} \tag{14}$$

The matrix $\boldsymbol{X}$ has three columns, and the true $\boldsymbol{\beta}$ is $[1 \quad -1 \quad 1]$. We can see that the true values of the autoregressive parameters are $\rho_1 = 0.1$, $\rho_2 = 0.2$, and $\rho_3 = 0.3$. The innovation variance is $20^2 = 400$. The sample size is 50.

The following simple program runs the regression (14) and finds the ML estimates of all the paramaters.

$\langle$*ar3test.ect*$\rangle\equiv$

```
sample 1 50

read ar3.dat y x1 x2 x3
gen X = colcat(1,x1,x2,x3)

set p = 3
set maxiter = 30
set tol = 1E-7

include arml.ect

quit
```

The parameter estimates are as follows. Note that the first component of $\boldsymbol{\beta}$ is the estimated constant.

```
  beta          rho
-9.617560    -0.010177
 0.975236     0.134410
-0.901390     0.202930
 0.967855
 s2 = 395.040158
```

Index of Code Segments

ar3test.ect
  definition, 15
arml.ect
  definition, 15
ARp function
  definition, 7
  use, 15
Compute AdA
  definition, 6
  use, 7
Compute dA
  definition, 6
  use, 6
Create dV
  definition, 5
  use, 7
Create YW
  definition, 4
  use, 7
derivative matrix of Omega
  definition, 5
  use, 6
diagonal of RHS
  definition, 6
  use, 6
final output
  definition, 14
  use, 15
first p observations
  definition, 9–12
  use, 14
Get dimensions of problem
  definition, 7
  use, 9
Get matrix A
  definition, 4
  use, 7
Initialisation of Om
  definition, 4
  use, 7
initialisation phase
  definition, 9
  use, 15
initialise AdA
  definition, 6
  use, 7
initialise beta
  definition, 7
  use, 9
initialise rho
  definition, 8
  use, 9
iterative loop
  definition, 14
  use, 15
Matrix of lagged residuals
  definition, 8
  use, 9
Parts of the artificial regression
  definition, 9
  use, 9
Regressors for beta
  definition, 9
  use, 14
Residuals and standard error
  definition, 8
  use, 9
rhs for derivatives of Omega
  definition, 5
  use, 6
RHS of derivative equation
  definition, 5
  use, 6
rt2 definition
  definition, 12
  use, 15
Solve Yule-Walker equations
  definition, 4
  use, 7
the artificial regression
  definition, 12
  use, 14
update residuals and main part of regressand
  definition, 13
  use, 14
update sigma
  definition, 14
  use, 14
Yule-Walker coefficient matrix
  definition, 4
  use, 7