Fitting a Seasonal ARIMA Model Using the NAG C Library

May 11, 2009

The Numerical Algorithms Group Ltd
Wilkinson House
Jordan Hill
Oxford
OX2 8DR
UK

Web Site:	www.nag.co.uk
General Inquires:	info@nag.co.uk
Technical Support:	support@nag.co.uk

One commonly used family of time series models are seasonal autoregressive integrated moving average (ARIMA) models. In an ARIMA model the time series $y_1, y_2, \ldots, y_n$, for $t = 1, 2, \ldots, n$, is assumed to follow:

$$\Delta^d \Delta_s^D y_t - c = w_t$$

where $\Delta^d \Delta_s^D y_t$ is the result of applying non-seasonal differencing of order d and seasonal differencing of seasonality s and order D to the series y_t. The differenced series is then of length $N = n - d'$, where $d' = d + (D \times s)$ is the generalised order of differencing. The scalar c is the expected value of the differenced series, and the series $w_1, w_2, \ldots, w_N$ follows a zero-mean stationary autoregressive moving average (ARMA) model defined by a pair of recurrence equations. These express w_t in terms of an uncorrelated series a_t, via an intermediate series e_t. The first equation describes the seasonal structure:

$$w_t = \Psi_1 w_{t-s} + \Psi_2 w_{t-2\times s} + \ldots + \Psi_P w_{t-P\times s} + e_t - \Theta_1 e_{t-s} - \Theta_2 e_{t-2\times s} - \ldots - \Theta_Q e_{t-Q\times s}$$

The second equation describes the non-seasonal structure. If the model is purely non-seasonal the first equation is redundant and e_t above is equated with w_t:

$$e_t = \psi_1 e_{t-1} + \psi_2 e_{t-2} + \ldots + \psi_p e_{t-p} + a_t - \theta_1 a_{t-1} - \theta_2 a_{t-2} - \ldots - \theta_q a_{t-q}$$

Whilst the NAG Fortran library has a routine specifically for fitting (`G13AEF`) and forecasting (`G13AJF`) from seasonal ARIMA models, the C library does not.

Both the Fortran and C libraries do have a routine for fitting a multi-input model, relating one output series to one or more input series (`G13BEF` and `g13bec` respectively). In a multi-input series the output series y_t, for $t = 1, 2, \ldots, n$, is assumed to be the sum of (unobserved) components $z_{i,t}$ which are due m input series $x_{i,t}$, for $i = 1, 2, \ldots, m$. A typical component z_t may be either

- a simple regression component, $z_t = \omega x_t$, or
- a transfer function model component which allows for the effect of lagged values of the variable, related to x_t by

$$z_t = \delta_1 z_{t-1} + \delta_2 z_{t-2} + \ldots + \delta_p z_{t-p} + \omega_0 x_{t-b} - \omega_1 x_{t-b-1} - \ldots - \omega_q x_{t-b-q}$$

The output series y_t is defined as

$$y_t = z_{1,t} + \ldots + z_{m,t} + n_t$$

where n_t is the error, or output noise component, and is assumed to follow a (possibly seasonal) ARIMA model. Therefore if there are no input series, that is $m = 0$, the multi-input model is equivalent to a seasonal ARIMA model.

This article gives a brief description of how to fit a seasonal ARIMA model using the C library routine `g13bec` (nag_tsa_multi_inp_model_estim), and how to forecast from such a model using the C library routine `g13bjc` (nag_tsa_multi_inp_model_forecast). The article should be read in conjunction with the documentation for these two routines. A full set of example source code, data and expected results is available from the NAG web site.

We are fitting an ARIMA model so there are no input series. This means that there is only a single series (y, called the output series in the routine documentation), so

```
nseries = 1;
```

We are also going to be dynamically allocating memory, so the stride parameter `tdxxy` can be set to the number of series, one in this case.

```
tdxxy = 1;
```

Even though there are no input series, we still need to allocate memory to the transfer function structure

```
nag_tsa_transf_orders(nseries, &transfv, NAGERR_DEFAULT);
```

but it does not need to be initialised. The rest of the parameters are fairly self explanatory, with the `arimav` structure holding the parameters describing the ARIMA model being fit, and the number of parameters, `npara` being set to

```
npara = arimav.p + arimav.q + arimav.bigp + arimav.bigq + nseries;
```

we can then allocate some memory

```
para = NAG_ALLOC(npara, double);
sd = NAG_ALLOC(npara, double);
xxy = NAG_ALLOC((nxxy+nfv)*tdxxy, double);
```

In the above code snippet `nfv` refers to the number of values we wish to forecast using `g13bjc` and `para`, `sd`, `xxy`, `npara`, `nxxy` and `tdxxy` refer to the parameters from `g13bec`. The array `xxy` has been defined as having length `(nxxy+nfv)*tdxxy` but, as described above, when fitting a seasonal ARIMA model the parameter `tdxxy` has a value of 1 and therefore doesn't need to appear in the formula. It has been included for clarity as this is how the size of `xxy` is defined in the routine documentation. Because we are forecasting `nfv` values from the ARIMA model the `xxy` array needs to be large enough to hold the original series, (`nxxy*tdxxy`), and the forecast values (`nfv*tdxxy`).

Once the rest of the input parameters have been populated the model parameters can be estimated using

```
nag_tsa_multi_inp_model_estim(&arimav, nseries, &transfv, para, npara,
                              nxxy, xxy, tdxxy, sd, &rss, &objf, &df,
                              G13_DEFAULT, NAGERR_DEFAULT);
```

Once the ARIMA model parameters have been estimated, the model can be used to produce forecasts by calling `g13bjc` (nag_tsa_multi_inp_model_forecast). The parameter `nev` is the number of observed values for the output series, and so

```
nev = nxxy;
```

There are no input series, so `rmsxy` need not be initialised, `mrx` and `parx` can be set to null and `tdmrx`, `ldparx` and `tdparx` to 1.

```
mrx = 0;
parx = 0;
tdmrx = ldparx = tdparx = 1;
```

Once the other input parameters have been initialised, the forecast can then be made using

```
nag_tsa_multi_inp_model_forecast(&arimav, nseries, &transfv, para,
                                 npara, nev, nfv, xxy, tdxxy, rmsxy,
                                 mrx, tdmrx, parx, ldparx, tdparx,
                                 fva, fsd, G13_DEFAULT, NAGERR_DEFAULT);
```