NAG Library Function Document

nag_tsa_varma_estimate (g13ddc)

1 Purpose

nag_tsa_varma_estimate (g13ddc) fits a vector autoregressive moving average (VARMA) model to an observed vector of time series using the method of Maximum Likelihood (ML). Standard errors of parameter estimates are computed along with their appropriate correlation matrix. The function also calculates estimates of the residual series.

2 Specification

```
#include <nag.h>
#include <nagg13.h>
void nag_tsa_varma_estimate (Integer k, Integer n, Integer ip, Integer iq,
     Nag_IncludeMean mean, double par[], Integer npar, double qq[],
     Integer kmax, const double w[], const Nag_Boolean parhld[],
     Nag_Boolean exact, Integer iprint, double cgetol, Integer maxcal,
     Integer ishow, const char *outfile, Integer *niter, double *rlogl,
     double v[], double g[], double cm[], Integer pdcm, NagError *fail)
```

3 Description

Let $W_t = (w_{1t}, w_{2t}, \ldots, w_{kt})^{\mathrm{T}}$, for $t = 1, 2, \ldots, n$, denote a vector of k time series which is assumed to follow a multivariate ARMA model of the form

$$\begin{aligned} W_t - \mu = \quad & \phi_1(W_{t-1} - \mu) + \phi_2(W_{t-2} - \mu) + \cdots + \phi_p\big(W_{t-p} - \mu\big) \\ & + \epsilon_t - \theta_1\epsilon_{t-1} - \theta_2\epsilon_{t-2} - \cdots - \theta_q\epsilon_{t-q} \end{aligned} \tag{1}$$

where $\epsilon_t = (\epsilon_{1t}, \epsilon_{2t}, \ldots, \epsilon_{kt})^{\mathrm{T}}$, for $t = 1, 2, \ldots, n$, is a vector of k residual series assumed to be Normally distributed with zero mean and positive definite covariance matrix Σ. The components of ϵ_t are assumed to be uncorrelated at non-simultaneous lags. The ϕ_i and θ_j are k by k matrices of parameters. $\{\phi_i\}$, for $i = 1, 2, \ldots, p$, are called the autoregressive (AR) parameter matrices, and $\{\theta_i\}$, for $i = 1, 2, \ldots, q$, the moving average (MA) parameter matrices. The parameters in the model are thus the p (k by k) ϕ-matrices, the q (k by k) θ-matrices, the mean vector, μ, and the residual error covariance matrix Σ. Let

$$A(\phi) = \begin{bmatrix} \phi_1 & I & 0 & . & . & . & 0 \\ \phi_2 & 0 & I & 0 & . & . & 0 \\ . & & & . & & & \\ . & & & & . & & \\ . & & & & & . & \\ \phi_{p-1} & 0 & . & . & . & 0 & I \\ \phi_p & 0 & . & . & . & 0 & 0 \end{bmatrix}_{pk \times pk} \quad \text{and} \quad B(\theta) = \begin{bmatrix} \theta_1 & I & 0 & . & . & . & 0 \\ \theta_2 & 0 & I & 0 & . & . & 0 \\ . & & & . & & & \\ . & & & & . & & \\ . & & & & & . & \\ \theta_{q-1} & 0 & . & . & . & . & I \\ \theta_q & 0 & . & . & . & . & 0 \end{bmatrix}_{qk \times qk}$$

where I denotes the k by k identity matrix.

The ARMA model (1) is said to be stationary if the eigenvalues of $A(\phi)$ lie inside the unit circle. Similarly, the ARMA model (1) is said to be invertible if the eigenvalues of $B(\theta)$ lie inside the unit circle.

The method of computing the exact likelihood function (using a Kalman filter algorithm) is discussed in Shea (1987). A quasi-Newton algorithm (see Gill and Murray (1972)) is then used to search for the maximum of the log-likelihood function. Stationarity and invertibility are enforced on the model using the reparameterisation discussed in Ansley and Kohn (1986). Conditional on the maximum likelihood estimates being equal to their true values the estimates of the residual series are uncorrelated with zero mean and constant variance Σ.

You have the option of setting an argument (**exact** to Nag_FALSE) so that nag_tsa_varma_estimate (g13ddc) calculates conditional maximum likelihood estimates (conditional on $W_0 = W_{-1} = \cdots = W_{1-p} = \epsilon_0 = \epsilon_{-1} = \cdots = \epsilon_{1-q} = 0$). This may be useful if the exact maximum likelihood estimates are close to the boundary of the invertibility region.

You also have the option (see Section 5) of requesting nag_tsa_varma_estimate (g13ddc) to constrain elements of the ϕ and θ matrices and μ vector to have pre-specified values.

4 References

Ansley C F and Kohn R (1986) A note on reparameterising a vector autoregressive moving average model to enforce stationarity *J. Statist. Comput. Simulation* **24** 99–106

Gill P E and Murray W (1972) Quasi-Newton methods for unconstrained optimization *J. Inst. Math. Appl.* **9** 91–108

Shea B L (1987) Estimation of multivariate time series *J. Time Ser. Anal.* **8** 95–110

5 Arguments

1: **k** – Integer *Input*

On entry: k, the number of observed time series.

Constraint: **k** ≥ 1.

2: **n** – Integer *Input*

On entry: n, the number of observations in each time series.

3: **ip** – Integer *Input*

On entry: p, the number of AR parameter matrices.

Constraint: **ip** ≥ 0.

4: **iq** – Integer *Input*

On entry: q, the number of MA parameter matrices.

Constraint: **iq** ≥ 0.

ip = **iq** = 0 is **not permitted**.

5: **mean** – Nag_IncludeMean *Input*

On entry: **mean** = Nag_MeanInclude, if components of μ have been estimated and **mean** = Nag_MeanZero, if all elements of μ are to be taken as zero.

Constraint: **mean** = Nag_MeanInclude or Nag_MeanZero.

6: **par**[**npar**] – double *Input/Output*

On entry: initial parameter estimates read in row by row in the order $\phi_1, \phi_2, \ldots, \phi_p$, $\theta_1, \theta_2, \ldots, \theta_q, \mu$.

Thus,

if **ip** > 0, **par**$[(l-1) \times k \times k + (i-1) \times k + j - 1]$ must be set equal to an initial estimate of the (i,j)th element of ϕ_l, for $l = 1, 2, \ldots, p$, $i = 1, 2, \ldots, k$ and $j = 1, 2, \ldots, k$;

if **iq** > 0, **par**$[p \times k \times k + (l-1) \times k \times k + (i-1) \times k + j - 1]$ must be set equal to an initial estimate of the (i,j)th element of θ_l, $l = 1, 2, \ldots, q$ and $i, j = 1, 2, \ldots, k$;

if **mean** = Nag_MeanInclude, **par**$[(p+q)\times k\times k+i-1]$ should be set equal to an initial estimate of the ith component of μ ($\mu(i)$). (If you set **par**$[(p+q)\times k\times k+i-1]$ to 0.0 then nag_tsa_varma_estimate (g13ddc) will calculate the mean of the ith series and use this as an initial estimate of $\mu(i)$.)

The first $p\times k\times k$ elements of **par** must satisfy the stationarity condition and the next $q\times k\times k$ elements of **par** must satisfy the invertibility condition.

If in doubt set all elements of **par** to 0.0.

On exit: if **fail.code** = NE_NOERROR or **fail.code** = NE_G13D_BOUND, NE_G13D_DERIV, NE_G13D_MAX_LOGLIK, NE_G13D_MAXCAL or NE_HESS_NOT_POS_DEF then all the elements of **par** will be overwritten by the latest estimates of the corresponding ARMA parameters.

7: **npar** – Integer *Input*

On entry: **npar** is the number of initial parameter estimates.

Constraints:

if **mean** = Nag_MeanZero, **npar** must be set equal to $(p+q)\times k\times k$;
if **mean** = Nag_MeanInclude, **npar** must be set equal to $(p+q)\times k\times k+k$.

The total number of observations $(n\times k)$ must exceed the total number of parameters in the model (**npar** $+k(k+1)/2$).

8: **qq**[**kmax** × **k**] – double *Input/Output*

On entry: **qq**$[\mathbf{kmax}\times(j-1)+i-1]$ must be set equal to an initial estimate of the (i,j)th element of Σ. The lower triangle only is needed. **qq** must be positive definite. It is strongly recommended that on entry the elements of **qq** are of the same order of magnitude as at the solution point. If you set **qq**$[\mathbf{kmax}\times(j-1)+i-1]=0.0$, for $i=1,2,\ldots,k$ and $j=1,2,\ldots,i$, then nag_tsa_varma_estimate (g13ddc) will calculate the covariance matrix between the k time series and use this as an initial estimate of Σ.

On exit: if **fail.code** = NE_NOERROR or **fail.code** = NE_G13D_BOUND, NE_G13D_DERIV, NE_G13D_MAX_LOGLIK, NE_G13D_MAXCAL or NE_HESS_NOT_POS_DEF then **qq**$[\mathbf{kmax}\times(j-1)+i-1]$ will contain the latest estimate of the (i,j)th element of Σ. The lower triangle only is returned.

9: **kmax** – Integer *Input*

On entry: stride seperating row elements in **qq**, **w** and **v**.

Constraint: **kmax** $\geq$ **k**.

10: **w**[**kmax** × **n**] – const double *Input*

On entry: **w**$[\mathbf{kmax}\times(t-1)+i-1]$ must be set equal to the ith component of W_t, for $i=1,2,\ldots,k$ and $t=1,2,\ldots,n$.

11: **parhld**[**npar**] – const Nag_Boolean *Input*

On entry: **parhld**$[i-1]$ must be set to Nag_TRUE if **par**$[i-1]$ is to be held constant at its input value and Nag_FALSE if **par**$[i-1]$ is a free parameter, for $i=1,2,\ldots,$ **npar**.

If in doubt try setting all elements of **parhld** to Nag_FALSE.

12: **exact** – Nag_Boolean *Input*

On entry: must be set equal to Nag_TRUE if you wish nag_tsa_varma_estimate (g13ddc) to compute exact maximum likelihood estimates. **exact** must be set equal to Nag_FALSE if only conditional likelihood estimates are required.

13: **iprint** – Integer *Input*

On entry: the frequency with which the automatic monitoring function is to be called.

iprint > 0
The ML search procedure is monitored once every **iprint** iterations and just before exit from the search function.

iprint $= 0$
The search function is monitored once at the final point.

iprint < 0
The search function is not monitored at all.

14: **cgetol** – double *Input*

On entry: the accuracy to which the solution in **par** and **qq** is required.

If **cgetol** is set to 10^{-l} and on exit **fail.code** = NE_NOERROR or **fail.code** = NE_G13D_BOUND, NE_G13D_DERIV or NE_HESS_NOT_POS_DEF, then all the elements in **par** and **qq** should be accurate to approximately l decimal places. For most practical purposes the value 10^{-4} should suffice. You should be wary of setting **cgetol** too small since the convergence criteria may then have become too strict for the machine to handle.

If **cgetol** has been set to a value which is less than the ***machine precision***, ϵ, then nag_tsa_varma_estimate (g13ddc) will use the value $10.0 \times \sqrt{\epsilon}$ instead.

15: **maxcal** – Integer *Input*

On entry: the maximum number of likelihood evaluations to be permitted by the search procedure.

Suggested value: **maxcal** $= 40 \times$ **npar** $\times$ (**npar** $+ 5$).

Constraint: **maxcal** ≥ 1.

16: **ishow** – Integer *Input*

On entry: specifies which of the following two quantities are to be printed.

(i) table of maximum likelihood estimates and their standard errors (as returned in the output arrays **par**, **qq** and **cm**);

(ii) table of residual series (as returned in the output array **v**).

ishow $= 0$
None of the above are printed.

ishow $= 1$
(i) only is printed.

ishow $= 2$
(i) and (ii) are printed.

Constraint: $0 \leq$ **ishow** ≤ 2.

17: **outfile** – const char * *Input*

On entry: the name of a file to which diagnostic output will be directed. If **outfile** is **NULL** the diagnostic output will be directed to standard output.

18: **niter** – Integer * *Output*

On exit: if **fail.code** = NE_NOERROR or **fail.code** = NE_G13D_BOUND, NE_G13D_DERIV, NE_G13D_MAX_LOGLIK, NE_G13D_MAXCAL or NE_HESS_NOT_POS_DEF then **niter** contains the number of iterations performed by the search function.

19: **rlogl** – double * *Output*

On exit: if **fail.code** = NE_NOERROR or **fail.code** = NE_G13D_BOUND, NE_G13D_DERIV, NE_G13D_MAX_LOGLIK, NE_G13D_MAXCAL or NE_HESS_NOT_POS_DEF then **rlogl** contains the value of the log-likelihood function corresponding to the final point held in **par** and **qq**.

20: **v**[**kmax** × **n**] – double *Output*

On exit: if **fail.code** = NE_NOERROR or **fail.code** = NE_G13D_BOUND, NE_G13D_DERIV, NE_G13D_MAX_LOGLIK, NE_G13D_MAXCAL or NE_HESS_NOT_POS_DEF then **v**[**kmax** × $(t-1)+i-1$] will contain an estimate of the ith component of ϵ_t, for $i = 1, 2, \ldots, k$ and $t = 1, 2, \ldots, n$, corresponding to the final point held in **par** and **qq**.

21: **g**[**npar**] – double *Output*

On exit: if **fail.code** = NE_NOERROR or **fail.code** = NE_G13D_BOUND, NE_G13D_DERIV, NE_G13D_MAX_LOGLIK, NE_G13D_MAXCAL or NE_HESS_NOT_POS_DEF then **g**$[i-1]$ will contain the estimated first derivative of the log-likelihood function with respect to the ith element in the array **par**. If the gradient cannot be computed then all the elements of **g** are returned as zero.

22: **cm**[**pdcm** × **npar**] – double *Output*

On exit: if **fail.code** = NE_NOERROR or **fail.code** = NE_G13D_BOUND, NE_G13D_DERIV, NE_G13D_MAX_LOGLIK, NE_G13D_MAXCAL or NE_HESS_NOT_POS_DEF then **cm**[**pdcm** × $(j-1)+i-1$] will contain an estimate of the correlation coefficient between the ith and jth elements in the **par** array for $1 \leq i \leq$ **npar**, $1 \leq j \leq$ **npar**. If $i = j$, then **cm**[**pdcm** × $(j-1)+i-1$] will contain the estimated standard error of **par**$[i-1]$. If the lth component of **par** has been held constant, i.e., **parhld**$[l-1]$ was set to Nag_TRUE, then the lth row and column of **cm** will be set to zero. If the second derivative matrix cannot be computed then all the elements of **cm** are returned as zero.

23: **pdcm** – Integer *Input*

On entry: stride seperating row elements in **cm**.

Constraint: **pdcm** ≥ **npar**.

24: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

NE_BAD_PARAM

On entry, argument ⟨*value*⟩ had an illegal value.

NE_G13D_AR

The initial AR parameter estimates are outside the stationarity region.
To proceed, you must try a different starting point.

NE_G13D_ARMA

On entry, **ip** = 0 and **iq** = 0.

NE_G13D_BOUND

The ML solution is so close to the boundary of either the stationarity region or the invertibility region that nag_tsa_varma_estimate (g13ddc) cannot evaluate the Hessian matrix. The elements of **cm** are set to zero, as are the elements of **g**. All other output quantities are correct.

NE_G13D_DERIV

An estimate of the second derivative matrix and the gradient vector at the solution point was computed. Either the Hessian matrix was found to be too ill-conditioned to be evaluated accurately or the gradient vector could not be computed to an acceptable degree of accuracy. The elements of **cm** are set to zero, as are the elements of **g**. All other output quantities are correct.

NE_G13D_GRAD

The function cannot compute a sufficiently accurate estimate of the gradient vector at the user-supplied starting point. This usually occurs if either the initial parameter estimates are very close to the ML parameter estimates, or you have supplied a very poor estimate of Σ, or the starting point is very close to the boundary of the stationarity or invertibility region. To proceed, you must try a different starting point.

NE_G13D_MA

The initial MA parameter estimates are outside the invertibility region. To proceed, you must try a different starting point.

NE_G13D_MAX_LOGLIK

The conditions for a solution have not all been met, but a point at which the log-likelihood took a larger value could not be found.

Provided that the estimated first derivatives are sufficiently small, and that the estimated condition number of the second derivative (Hessian) matrix, as printed when **iprint** ≥ 0*, is not too large, this error exit may simply mean that, although it has not been possible to satisfy the specified requirements, the algorithm has in fact found the solution as far as the accuracy of the machine permits.*

Such a condition can arise, for instance, if ***cgetol*** *has been set so small that rounding error in evaluating the likelihood function makes attainment of the convergence conditions impossible.*

If the estimated condition number at the final point is large, it could be that the final point is a solution but that the smallest eigenvalue of the Hessian matrix is so close to zero at the solution that it is not possible to recognize it as a solution. Output quantities were computed at the final point held in ***par*** *and* ***qq****, except that if* ***g*** *or* ***cm*** *could not be computed, in which case they are set to zero.*

NE_G13D_MAXCAL

There have been **maxcal** log-likelihood evaluations made in the function.

If steady increases in the log-likelihood function were monitored up to the point where this exit occurred, then the exit probably simply occurred because ***maxcal*** *was set too small, so the calculations should be restarted from the final point held in* ***par*** *and* ***qq****. This type of exit may also indicate that there is no maximum to the likelihood surface. Output quantities were computed at the final point held in* ***par*** *and* ***qq****, except that if* ***g*** *or* ***cm*** *could not be computed, in which case they are set to zero.*

NE_G13D_START

The starting point is too close to the boundary of the admissibility region.

NE_HESS_NOT_POS_DEF

The second-derivative matrix at the solution point is not positive definite. The elements of **cm** are set to zero. All other output quantities are correct.

NE_INT

On entry, **ip** = ⟨*value*⟩.
Constraint: **ip** ≥ 0.

On entry, **iq** = ⟨*value*⟩.
Constraint: **iq** ≥ 0.

On entry, **ishow** = ⟨*value*⟩.
Constraint: $0 \leq$ **ishow** ≤ 2.

On entry, **k** = ⟨*value*⟩.
Constraint: **k** ≥ 1.

On entry, **maxcal** = ⟨*value*⟩.
Constraint: **maxcal** ≥ 1.

On entry, **npar** = ⟨*value*⟩.
Constraint: **npar** = ⟨*value*⟩.

On entry, **npar** = ⟨*value*⟩.
Constraint: **npar** ≥ 0.

NE_INT_2

On entry, **kmax** = ⟨*value*⟩ and **k** = ⟨*value*⟩.
Constraint: **kmax** $\geq$ **k**.

On entry, **pdcm** = ⟨*value*⟩ and **npar** = ⟨*value*⟩.
Constraint: **pdcm** $\geq$ **npar**.

NE_INT_3

On entry, **n** = ⟨*value*⟩, **k** = ⟨*value*⟩ and **npar** = ⟨*value*⟩.
Constraint: **n** $\times$ **k** $>$ **npar** $+$ **k** $\times$ (**k** $+ 1$)/2.

NE_INTERNAL_ERROR

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please contact NAG for assistance.

An unexpected error has been triggered by this function. Please contact NAG.
See Section 3.6.6 in the Essential Introduction for further information.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.
See Section 3.6.5 in the Essential Introduction for further information.

NE_NOT_CLOSE_FILE

Cannot close file ⟨*value*⟩.

NE_NOT_POS_DEF

The initial estimate of Σ is not positive definite. To proceed, you must try a different starting point.

NE_NOT_WRITE_FILE

Cannot open file ⟨*value*⟩ for writing.

7 Accuracy

On exit from nag_tsa_varma_estimate (g13ddc), if **fail**.**code** = NE_NOERROR or **fail**.**code** = NE_G13D_BOUND, NE_G13D_DERIV or NE_HESS_NOT_POS_DEF and **cgetol** has been set to 10^{-l}, then all the parameters should be accurate to approximately l decimal places. If **cgetol** was set equal to a value less than the ***machine precision***, ϵ, then all the parameters should be accurate to approximately $10.0 \times \sqrt{\epsilon}$.

If **fail**.**code** = NE_G13D_MAXCAL on exit (i.e., **maxcal** likelihood evaluations have been made but the convergence conditions of the search function have not been satisfied), then the elements in **par** and **qq** may still be good approximations to the ML estimates. Inspection of the elements of **g** may help you determine whether this is likely.

8 Parallelism and Performance

nag_tsa_varma_estimate (g13ddc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_tsa_varma_estimate (g13ddc) makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this function. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

9.1 Memory Usage

Let $r = \max(\mathbf{ip}, \mathbf{iq})$ and $s = \mathbf{npar} + \mathbf{k} \times (\mathbf{k} + 1)/2$. Local workspace arrays of fixed lengths are allocated internally by nag_tsa_varma_estimate (g13ddc). The total size of these arrays amounts to $s + \mathbf{k} \times r + 52$ Integer elements and
$2 \times s^2 + s \times (s - 1)/2 + 15 \times s + \mathbf{k}^2 \times \left(2 \times \mathbf{ip} + \mathbf{iq} + (r + 3)^2\right) + \mathbf{k} \times (2 \times r^2 + 2 \times$
$r + 3 \times \mathbf{n} + 4) + 10$ double elements.

9.2 Timing

The number of iterations required depends upon the number of parameters in the model and the distance of the user-supplied starting point from the solution.

9.3 Constraining for Stationarity and Invertibility

If the solution lies on the boundary of the admissibility region (stationarity and invertibility region) then nag_tsa_varma_estimate (g13ddc) may get into difficulty and exit with **fail**.**code** = NE_G13D_MAX_LOGLIK. If this exit occurs you are advised to either try a different starting point or a different setting for **exact**. If this still continues to occur then you are urged to try fitting a more parsimonious model.

9.4 Over-parameterisation

You are advised to try and avoid fitting models with an excessive number of parameters since over-parameterisation can cause the maximization problem to become ill-conditioned.

9.5 Standardizing the Residual Series

The standardized estimates of the residual series ϵ_t (denoted by $\hat{e}_t$) can easily be calculated by forming the Cholesky decomposition of Σ, e.g., GG^{T} and setting $\hat{e}_t = G^{-1}\hat{\epsilon}_t$. nag_dpotrf (f07fdc) may be used to calculate the array **g**. The components of $\hat{e}_t$ which are now uncorrelated at **all** lags can sometimes be more easily interpreted.

9.6 Assessing the Fit of the Model

If your time series model provides a good fit to the data then the residual series should be approximately white noise, i.e., exhibit no serial cross-correlation. An examination of the residual cross-correlation matrices should confirm whether this is likely to be so. You are advised to call nag_tsa_varma_diagnostic (g13dsc) to provide information for diagnostic checking. nag_tsa_varma_diagnostic (g13dsc) returns the residual cross-correlation matrices along with their asymptotic standard errors. nag_tsa_varma_diagnostic (g13dsc) also computes a portmanteau statistic and its asymptotic significance level for testing model adequacy. If **fail**.**code** = NE_NOERROR or **fail**.**code** = NE_G13D_BOUND, NE_G13D_DERIV, NE_G13D_MAX_LOGLIK or NE_HESS_NOT_POS_DEF on exit from nag_tsa_varma_estimate (g13ddc) then the quantities output **k**, **n**, **v**, **kmax**, **ip**, **iq**, **par**, **parhld**, and **qq** will be suitable for input to nag_tsa_varma_diagnostic (g13dsc).

10 Example

This example shows how to fit a bivariate AR(1) model to two series each of length 48. μ will be estimated and $\phi_1(2,1)$ will be constrained to be zero.

10.1 Program Text

```
/* nag_tsa_varma_estimate (g13ddc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 9, 2009.
 */
/* Pre-processor includes */
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg13.h>

int main(void)
{
  /*Logical scalar and array declarations */
  Nag_Boolean     exact;
  Nag_IncludeMean mean;
  Nag_Boolean     *parhld = 0;
  /*Integer scalar and array declarations */
  Integer         exit_status = 0;
  Integer         i, ip, iprint, iq, ishow, j, k, kmax, maxcal, n;
  Integer         niter, npar, pdcm, pdqq, pdw;
  /*Double scalar and array declarations */
  double          cgetol, rlogl;
  double          *cm = 0, *g = 0, *par = 0, *qq = 0, *v = 0, *w = 0;
  /*Character scalar and array declarations */
  char            smean[40];
  /*NAG Types */
  NagError        fail;

  INIT_FAIL(fail);

  printf("nag_tsa_varma_estimate (g13ddc) Example Program Results\n");
  /*     Skip heading in data file*/
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif
#ifdef _WIN32
  scanf_s("%"NAG_IFMT"%"NAG_IFMT"%"NAG_IFMT"%"NAG_IFMT"%39s%*[^\n] ", &k,
        &ip, &iq, &n, smean, _countof(smean));
#else
  scanf("%"NAG_IFMT"%"NAG_IFMT"%"NAG_IFMT"%"NAG_IFMT"%39s%*[^\n] ", &k,
```

```
        &ip, &iq, &n, smean);
#endif
  npar = (ip+iq)*k*k;
  kmax = 3;
  /*
   * nag_enum_name_to_value (x04nac).
   * Converts NAG enum member name to value
   */
  mean = (Nag_IncludeMean) nag_enum_name_to_value(smean);
  if (mean == Nag_MeanInclude)
    npar = npar+k;
  printf("\n");
  pdcm = npar;
    #define CM(I, J) cm[(J-1)*pdcm + I-1]
  pdqq = kmax;
    #define QQ(I, J) qq[(J-1)*pdqq + I-1]
  pdw = kmax;
    #define W(I, J)  w[(J-1)*pdw + I-1]
  if (!(cm = NAG_ALLOC(npar*npar, double)) ||
      !(g = NAG_ALLOC(npar, double)) ||
      !(par = NAG_ALLOC(npar, double)) ||
      !(qq = NAG_ALLOC(kmax*k, double)) ||
      !(v = NAG_ALLOC(kmax*n, double)) ||
      !(w = NAG_ALLOC(kmax*n, double)) ||
      !(parhld = NAG_ALLOC(npar, Nag_Boolean)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }
  for (i = 0; i < npar; i++)
    {
      par[i] = 0.00e0;
      parhld[i] = Nag_FALSE;
    }
  /*     Set all elements of Q to zero to use the covariance matrix */
  /*     between the K time series as the initial estimate of the   */
  /*     covariance matrix                                          */
  for (j = 1; j <= k; j++)
    {
      for (i = j; i <= k; i++)
        QQ(i, j) = 0.00e0;
    }
  for (i = 1; i <= k; i++)
    {
      for (j = 1; j <= n; j++)
#ifdef _WIN32
        scanf_s("%lf ", &W(i, j));
#else
        scanf("%lf ", &W(i, j));
#endif
    }
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif
  parhld[2] = Nag_TRUE;
  exact = Nag_TRUE;
  iprint = (-(1));
  cgetol = 0.00010e0;
  maxcal = 40*npar*(npar+5);
  ishow = 2;
  /*
   * nag_tsa_varma_estimate (g13ddc)
   * Multivariate time series, estimation of varma model
   */
  fflush(stdout);
  nag_tsa_varma_estimate(k, n, ip, iq, mean, par, npar, qq, kmax, w, parhld,
                         exact, iprint, cgetol, maxcal, ishow, 0,
                         &niter, &rlogl, v, g, cm, pdcm, &fail);
```

```
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_tsa_varma_estimate (g13ddc).\n%s\n",
              fail.message);
      exit_status = 1;
      goto END;
    }

 END:
  NAG_FREE(cm);
  NAG_FREE(g);
  NAG_FREE(par);
  NAG_FREE(qq);
  NAG_FREE(v);
  NAG_FREE(w);
  NAG_FREE(parhld);

  return exit_status;
}
```

10.2 Program Data

```
nag_tsa_varma_estimate (g13ddc) Example Program Data
2 1 0 48 Nag_MeanInclude                                  : k, ip, iq, n, mean
-1.490 -1.620  5.200  6.230  6.210  5.860  4.090  3.180
 2.620  1.490  1.170  0.850 -0.350  0.240  2.440  2.580
 2.040  0.400  2.260  3.340  5.090  5.000  4.780  4.110
 3.450  1.650  1.290  4.090  6.320  7.500  3.890  1.580
 5.210  5.250  4.930  7.380  5.870  5.810  9.680  9.070
 7.290  7.840  7.550  7.320  7.970  7.760  7.000  8.350
 7.340  6.350  6.960  8.540  6.620  4.970  4.550  4.810
 4.750  4.760 10.880 10.010 11.620 10.360  6.400  6.240
 7.930  4.040  3.730  5.600  5.350  6.810  8.270  7.680
 6.650  6.080 10.250  9.140 17.750 13.300  9.630  6.800
 4.080  5.060  4.940  6.650  7.940 10.760 11.890  5.850
 9.010  7.500 10.020 10.380  8.150  8.370 10.730 12.140 : w
```

10.3 Program Results

```
nag_tsa_varma_estimate (g13ddc) Example Program Results


 VALUE OF LOG LIKELIHOOD FUNCTION ON EXIT = -0.20280E+03

 MAXIMUM LIKELIHOOD ESTIMATES OF AR PARAMETER MATRICES
 -----------------------------------------------------

 PHI(1)    ROW-WISE :    0.802   0.065
                        (0.091) (0.102)

                         0.000   0.575
                        (0.000) (0.121)

 MAXIMUM LIKELIHOOD ESTIMATE OF PROCESS MEAN
 -------------------------------------------

                         4.271   7.825
                        (1.219) (0.776)

 MAXIMUM LIKELIHOOD ESTIMATE OF SIGMA MATRIX
 -------------------------------------------

                         2.964

                         0.637   5.380

           RESIDUAL SERIES NUMBER  1
           ------------------------

   T     1      2      3      4      5      6      7      8
```

```
 V(T)  -3.33  -1.24   5.75   1.27   0.32   0.11  -1.27  -0.73

   T     9     10     11     12     13     14     15     16
 V(T)  -0.58  -1.26  -0.67  -1.13  -2.02  -0.57   1.24  -0.13

   T    17     18     19     20     21     22     23     24
 V(T)  -0.77  -2.09   1.34   0.95   1.71   0.23  -0.01  -0.60

   T    25     26     27     28     29     30     31     32
 V(T)  -0.68  -1.89  -0.77   2.05   2.11   0.94  -3.32  -2.50

   T    33     34     35     36     37     38     39     40
 V(T)   3.16   0.47   0.05   2.77  -0.82   0.25   3.99   0.20

   T    41     42     43     44     45     46     47     48
 V(T)  -0.70   1.07   0.44   0.28   1.09   0.50  -0.10   1.70

           RESIDUAL SERIES NUMBER  2
           -------------------------

   T     1      2      3      4      5      6      7      8
 V(T)  -0.19  -1.20  -0.02   1.21  -1.62  -2.16  -1.63  -1.13

   T     9     10     11     12     13     14     15     16
 V(T)  -1.34  -1.30   4.82   0.43   2.54   0.35  -2.88  -0.77

   T    17     18     19     20     21     22     23     24
 V(T)   1.02  -3.85  -1.92   0.13  -1.20   0.41   1.03  -0.40

   T    25     26     27     28     29     30     31     32
 V(T)  -1.09  -1.07   3.43  -0.08   9.17  -0.23  -1.34  -2.06

   T    33     34     35     36     37     38     39     40
 V(T)  -3.16  -0.61  -1.30   0.48   0.79   2.87   2.38  -4.31

   T    41     42     43     44     45     46     47     48
 V(T)   2.32  -1.01   2.38   1.29  -1.14   0.36   2.59   2.64
```