

# NAG C 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 argument 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 ik,
    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 icm, NagError *fail)
```

### 3 Description

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

$$W_t - \mu = \phi_1(W_{t-1} - \mu) + \phi_2(W_{t-2} - \mu) + \dots + \phi_p(W_{t-p} - \mu) + \epsilon_t - \theta_1\epsilon_{t-1} - \theta_2\epsilon_{t-2} - \dots - \theta_q\epsilon_{t-q} \quad (1)$$

where  $\epsilon_t = (\epsilon_{1t}, \epsilon_{2t}, \dots, \epsilon_{kt})^T$ , for  $t = 1, 2, \dots, n$  is a vector of  $k$  residual series assumed to be Normally distributed with zero mean and positive-definite covariance matrix  $\Sigma$ . The components of  $\epsilon_t$  are assumed to be uncorrelated at non-simultaneous lags. The  $\phi_i$  and  $\theta_j$  are  $k$  by  $k$  matrices of arguments.  $\{\phi_i\}$ , for  $i = 1, 2, \dots, p$ , are called the autoregressive (AR) argument matrices, and  $\{\theta_i\}$ , for  $i = 1, 2, \dots, q$ , the moving average (MA) argument matrices. The arguments in the model are thus the  $p$  ( $k$  by  $k$ )  $\phi$ -matrices, the  $q$  ( $k$  by  $k$ )  $\theta$ -matrices, the mean vector,  $\mu$ , and the residual error covariance matrix  $\Sigma$ . Let

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

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  $\Sigma$ .

You have the option of setting a argument (**exact** to **Nag\_False**) so that `nag_tsa_varma_estimate` (g13ddc) calculates conditional maximum likelihood estimates (conditional on  $W_0 = W_{-1} = \dots = W_{1-p} = \epsilon_0 = \epsilon_{-1} = \dots = \epsilon_{1-q} = 0$ ). 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  $\phi$  and  $\theta$  matrices and  $\mu$  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:*  $k \geq 1$ .
- 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 argument matrices.  
*Constraint:*  $ip \geq 0$ .
- 4: **iq** – Integer *Input*  
*On entry:*  $q$ , the number of MA argument matrices.  
*Constraint:*  $iq \geq 0$ .  
**ip = iq = 0 is not permitted.**
- 5: **mean** – Nag\_IncludeMean *Input*  
*On entry:* **mean = Nag\_MeanInclude**, if components of  $\mu$  have been estimated and **mean = Nag\_MeanZero**, if all elements of  $\mu$  are to be taken as zero.  
*Constraint:* **mean = Nag\_MeanInclude** or **Nag\_MeanZero**.
- 6: **par[npar]** – double *Input/Output*  
*On entry:* initial argument estimates read in row by row in the order  $\phi_1, \phi_2, \dots, \phi_p, \theta_1, \theta_2, \dots, \theta_q, \mu$ . Thus, if  $ip > 0$  then **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  $\phi_l$ , for  $l = 1, 2, \dots, p$ ;  $i, j = 1, 2, \dots, k$ . If  $iq > 0$ , then **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  $\theta_l$ ,  $l = 1, 2, \dots, q$ ;  $i, j = 1, 2, \dots, k$ . If **mean = Nag\_MeanInclude**, then **par**[( $p+q$ )  $\times$   $k \times k + i - 1$ ] should be set equal to an initial estimate of the  $i$ th component of  $\mu$  ( $\mu(i)$ ). (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  $i$ th 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\_HESS\_NOT\_POS\_DEF**, **NE\_G13D\_BOUND**, **NE\_G13D\_DERIV**, **NE\_G13D\_MAXCAL** or **NE\_G13D\_MAX\_LOGLIK** then all the elements of **par** will be overwritten by the latest estimates of the corresponding ARMA arguments.

7: **npar** – Integer *Input*

*On entry:* the dimension of the arrays **par**, **parhld** and **g** and the second dimension of the array **cm** as declared in the function from which `nag_tsa_varma_estimate` (g13ddc) is called. The number of initial argument estimates. The total number of observations ( $n \times k$ ) must exceed the total number of arguments in the model ( $\mathbf{npar} + k(k + 1)/2$ ).

*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  $(p + q) \times k \times k + k$ .

8: **qq**[*dim*] – double *Input/Output*

**Note:** the dimension, *dim*, of the array **qq** must be at least  $\mathbf{k} \times \mathbf{ik}$ .

*On entry:* **qq**[ $\mathbf{ik} \times (j - 1) + i - 1$ ] must be set equal to an initial estimate of the (*i, j*)th element of  $\Sigma$ . 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**[ $\mathbf{ik} \times (j - 1) + i - 1$ ] = 0.0, for  $i = 1, 2, \dots, k$ ;  $j = 1, 2, \dots, i$ , then `nag_tsa_varma_estimate` (g13ddc) will calculate the covariance matrix between the  $k$  time series and use this as an initial estimate of  $\Sigma$ .

*On exit:* if **fail.code** = **NE\_NOERROR** or **fail.code** = **NE\_HESS\_NOT\_POS\_DEF**, **NE\_G13D\_BOUND**, **NE\_G13D\_DERIV**, **NE\_G13D\_MAXCAL** or **NE\_G13D\_MAX\_LOGLIK** then **qq**[ $\mathbf{ik} \times (j - 1) + i - 1$ ] will contain the latest estimate of the (*i, j*)th element of  $\Sigma$ . The lower triangle only is returned.

9: **ik** – Integer *Input*

*On entry:* the second dimension of the arrays **qq**, **w** and **v** as declared in the function from which `nag_tsa_varma_estimate` (g13ddc) is called.

*Constraint:*  $\mathbf{ik} \geq \mathbf{k}$ .

10: **w**[*dim*] – const double *Input*

**Note:** the dimension, *dim*, of the array **w** must be at least  $\mathbf{n} \times \mathbf{ik}$ .

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

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 argument, for  $i = 1, 2, \dots, \mathbf{npar}$ .

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 the function 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. See Section 9 for an example of the printed output.

**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\_HESS\_NOT\_POS\_DEF**, **NE\_G13D\_BOUND** or **NE\_G13D\_DERIV**, 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*,  $\epsilon$ , 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.

*Constraint:* **maxcal**  $\geq 1$ .

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

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:*  $0 \leq \text{ishow} \leq 2$ .

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\_HESS\_NOT\_POS\_DEF**, **NE\_G13D\_BOUND**, **NE\_G13D\_DERIV**, **NE\_G13D\_MAXCAL** or **NE\_G13D\_MAX\_LOGLIK** 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\_HESS\_NOT\_POS\_DEF**, **NE\_G13D\_BOUND**, **NE\_G13D\_DERIV**, **NE\_G13D\_MAXCAL** or **NE\_G13D\_MAX\_LOGLIK** then **rlogl** contains the value of the log-likelihood function corresponding to the final point held in **par** and **qq**.
- 20: **v[dim]** – double *Output*  
**Note:** the dimension, *dim*, of the array **w** must be at least  $\mathbf{n} \times \mathbf{ik}$ .  
*On exit:* if **fail.code** = **NE\_NOERROR** or **fail.code** = **NE\_HESS\_NOT\_POS\_DEF**, **NE\_G13D\_BOUND**, **NE\_G13D\_DERIV**, **NE\_G13D\_MAXCAL** or **NE\_G13D\_MAX\_LOGLIK** then **v[ik × (t - 1) + i - 1]** will contain an estimate of the *i*th component of  $\epsilon_t$ , for  $i = 1, 2, \dots, k$ ;  $t = 1, 2, \dots, n$ , 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\_HESS\_NOT\_POS\_DEF**, **NE\_G13D\_BOUND**, **NE\_G13D\_DERIV**, **NE\_G13D\_MAXCAL** or **NE\_G13D\_MAX\_LOGLIK** then **g[i - 1]** will contain the estimated first derivative of the log-likelihood function with respect to the *i*th element in the array **par**. If the gradient cannot be computed then all the elements of **g** are returned as zero.
- 22: **cm[icm × npar]** – double *Output*  
*On exit:* if **fail.code** = **NE\_NOERROR** or **fail.code** = **NE\_HESS\_NOT\_POS\_DEF**, **NE\_G13D\_BOUND**, **NE\_G13D\_DERIV**, **NE\_G13D\_MAXCAL** or **NE\_G13D\_MAX\_LOGLIK** then **cm[icm × (j - 1) + i - 1]** will contain an estimate of the correlation coefficient between the *i*th and *j*th elements in the **par** array for  $1 \leq i \leq \mathbf{npar}$ ,  $1 \leq j \leq \mathbf{npar}$ . If  $i = j$ , then **cm[icm × (j - 1) + i - 1]** will contain the estimated standard error of **par[i - 1]**. If the *l*th component of **par** has been held constant, i.e., **parhld[l - 1]** was set to **Nag\_True**, then the *l*th 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: **icm** – Integer *Input*  
*On entry:* the first dimension of the array **cm** as declared in the function from which **nag\_tsa\_varma\_estimate** (g13ddc) is called.  
**Constraint:** **icm**  $\geq$  **npar**.
- 24: **fail** – NagError \* *Input/Output*  
The NAG error argument (see Section 2.6 of the Essential Introduction).

## 6 Error Indicators and Warnings

### NE\_ALLOC\_FAIL

Dynamic memory allocation failed.

### NE\_BAD\_PARAM

On entry, argument  $\langle \text{value} \rangle$  had an illegal value.

### NE\_G13D\_AR

On entry, the AR argument estimates are outside the stationarity region.

### NE\_G13D\_ARMA

On entry, **ip** and **iq** are both zero.

**NE\_G13D\_BOUND**

The solution is too close to the admissibility boundary for standard errors and correlations of argument estimates to be calculated.

**NE\_G13D\_DERIV**

The elements of the 2nd derivative matrix and the gradient vector could not be evaluated accurately.

**NE\_G13D\_GRAD**

Minimization function cannot compute accurate estimate of gradient vector.

**NE\_G13D\_MA**

On entry, the MA argument estimates are outside the invertibility region.

**NE\_G13D\_MAXCAL**

**maxcal** likelihood evaluations have been made: **maxcal** =  $\langle value \rangle$ .

**NE\_G13D\_MAX\_LOGLIK**

Conditions for a solution have not all been met.

**NE\_G13D\_START**

Starting point is too close to boundary of admissibility region.

**NE\_HESS\_NOT\_POS\_DEF**

The Hessian is not positive-definite at the solution point.

**NE\_INT**

On entry, **ip** =  $\langle value \rangle$ .

Constraint: **ip**  $\geq 0$ .

On entry, **iq** =  $\langle value \rangle$ .

Constraint: **iq**  $\geq 0$ .

On entry, **ishow** is not equal to 0, 1, or 2: **ishow** =  $\langle value \rangle$ .

On entry, **k** =  $\langle value \rangle$ .

Constraint: **k**  $\geq 1$ .

On entry, **maxcal** =  $\langle value \rangle$ .

Constraint: **maxcal**  $\geq 1$ .

On entry, **npar** is not equal to  $\langle value \rangle$ : **npar** =  $\langle value \rangle$ .

**NE\_INT\_2**

On entry, **icm** < **npar**: **icm** =  $\langle value \rangle$ , **npar** =  $\langle value \rangle$ .

On entry, **ik** < **k**: **ik** =  $\langle value \rangle$ , **k** =  $\langle value \rangle$ .

**NE\_INT\_3**

On entry,  $\mathbf{n} \times \mathbf{k} \leq \mathbf{npar} + \mathbf{k} \times (\mathbf{k} + 1)/2$ : **n** =  $\langle value \rangle$ , **k** =  $\langle value \rangle$ , **npar** =  $\langle value \rangle$ .

**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 consult NAG for assistance.

**NE\_NOT\_CLOSE\_FILE**

Cannot close file  $\langle value \rangle$ .

**NE\_NOT\_POS\_DEF**

Initial estimate of  $\Sigma$  is not positive-definite.

**NE\_NOT\_WRITE\_FILE**

Cannot open file  $\langle value \rangle$  for writing.

**7 Accuracy**

On exit from `nag_tsa_varma_estimate` (g13ddc), if `fail.code = NE_NOERROR` or `fail.code = -NE_HESS_NOT_POS_DEF`, `NE_G13D_BOUND` or `NE_G13D_DERIV` and `cgetol` has been set to  $10^{-l}$ , then all the arguments should be accurate to approximately  $l$  decimal places. If `cgetol` was set equal to a value less than the *machine precision*,  $\epsilon$ , then all the arguments 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. You are advised to inspect the elements of `g` to see whether this is likely to be so.

**8 Further Comments****8.1 Timing**

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

**8.2 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.

**8.3 Over-parameterisation**

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

**8.4 Standardizing the Residual Series**

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

**8.5 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. The function also computes a portmanteau statistic and its asymptotic significance level for testing model adequacy. If `fail.code = NE_NOERROR` or `fail.code = NE_HESS_NOT_POS_DEF`, `NE_G13D_BOUND`,

**NE\_G13D\_DERIV** or **NE\_G13D\_MAX\_LOGLIK** on exit from `nag_tsa_varma_estimate` (g13ddc) then the output quantities **k**, **n**, **v**, **ik**, **ip**, **iq**, **par**, **parhld**, and **qq** will be suitable for input to `nag_tsa_varma_diagnostic` (g13dsc).

## 9 Example

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

### 9.1 Program Text

```

/* nag_tsa_varma_estimate (g13ddc) Example Program.
 *
 * Copyright 2005 Numerical Algorithms Group.
 *
 * Mark 8, 2004.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg13.h>

int main(void)
{
    /* Scalars */
    double cgetol, rlogl;
    Integer exit_status, i, icm, ik, ip, iprint, iq, ishow, j, k, maxcal, n;
    Integer niter, npar;
    Nag_Boolean exact;
    Nag_IncludeMean mean;
    char meanc;

    /* Arrays */
    char *outfile = 0;
    double *cm = 0, *g = 0, *par = 0, *qq = 0, *v = 0, *w = 0;
    /* Nag types */
    Nag_Boolean *parhld = 0;
    NagError fail;

#define W(I,J) w[(J - 1)*ik + I - 1]
#define QQ(I,J) qq[(J - 1)*ik + I - 1]

    INIT_FAIL(fail);
    exit_status = 0;

    Vprintf("nag_tsa_varma_estimate (g13ddc) Example Program Results\n\n");
    /* Skip heading in data file */
    Vscanf("%*[\n]");
    Vscanf("%ld%ld%ld%ld %c%*[\n]",
           &k, &ip, &iq, &n, &meanc);

    mean = Nag_MeanZero;
    if (meanc == 'm' || meanc == 'M')
    {
        mean = Nag_MeanInclude;
    }

    if ( k >= 1 && ip >= 0 && iq >= 0 && n >= 3)
    {
        /* Allocate memory */
        ik = k;
        if ( !(qq = NAG_ALLOC(k*ik, double)) ||
            !(v = NAG_ALLOC(n*ik, double)) ||
            !(w = NAG_ALLOC(n*ik, double)) )
        {
            Vprintf("Allocation failure.\n");
            exit_status = -1;

```

```

        goto END;
    }
}
else
{
    Vprintf("Invalid k, ip, iq or n.\n");
    exit_status = 1;
    goto END;
}

npar = (ip + iq) * k * k;
if (mean)
{
    npar += k;
}

/* Allocate memory */
icm = npar;
if ( !(cm = NAG_ALLOC(icm * npar, double)) ||
    !(g = NAG_ALLOC(npar, double)) ||
    !(par = NAG_ALLOC(npar, double)) ||
    !(parhld = NAG_ALLOC(npar, Nag_Boolean)) )
{
    Vprintf("Allocation failure.\n");
    exit_status = -1;
    goto END;
}

for (i = 1; i <= npar; ++i)
{
    par[i-1] = 0.0;
    parhld[i-1] = Nag_FALSE;
}
parhld[2] = Nag_TRUE;

/* Set all elements of Q to zero to use 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) = 0.0;
    }
}

for (i = 1; i <= k; ++i)
{
    for (j = 1; j <= n; ++j)
    {
        Vscanf("%lf", &W(i,j));
    }
}

exact = Nag_TRUE;

/* ** Set iprint > 0 to obtain intermediate output */
iprint = -1;
cgetol = 1.0e-4;
maxcal = npar * 40 * (npar + 5);
ishow = 2;

/* nag_tsa_varma_estimate (g13ddc).
 * Multivariate time series, estimation of VARMA model
 */
nag_tsa_varma_estimate(k, n, ip, iq, mean, par, npar, qq, ik, w, parhld,
    exact, iprint, cgetol, maxcal, ishow, outfile, &niter,
    &rlogl, v, g, cm, icm, &fail);

if (fail.code != NE_NOERROR)

```

```

    {
        vprintf("\n nag_tsa_varma_estimate (g13ddc) message: %s\n\n",
                fail.message);
    }

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

    return exit_status;
}

```

## 9.2 Program Data

```

nag_tsa_varma_estimate (g13ddc) Example Program Data
2 1 0 48 M
-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

```

## 9.3 Program Results

```

nag_tsa_varma_estimate (g13ddc) Example Program Results

VALUE OF IFAIL PARAMETER ON EXIT FROM G13DCF =    0

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

---