NAG Library Function Document

nag_tsa_multi_inp_model_estim (g13bec)

1 Purpose

nag_tsa_multi_inp_model_estim (g13bec) fits a time series model to one output series relating it to any input series with a choice of three different estimation criteria – nonlinear least squares, exact likelihood and marginal likelihood. When no input series are present, nag_tsa_multi_inp_model_estim (g13bec) fits a univariate ARIMA model.

2 Specification

```
#include <nag.h>
#include <nagg13.h>
```

```
void nag_tsa_multi_inp_model_estim (Nag_ArimaOrder *arimav, Integer nseries,
     Nag_TransfOrder *transfv, double para[], Integer npara, Integer nxxy,
     const double xxy[], Integer tdxxy, double sd[], double *rss,
     double *objf, double *df, Nag_G13_Opt *options, NagError *fail)
```

3 Description

3.1 The Multi-input Model

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 respectively to the inputs $x_{i,t}$, for $i = 1, 2, \ldots, m$.

Thus $y_t = z_{1,t} + \cdots + z_{m,t} + n_t$ where n_t is the error, or output noise component.

A typical component z_t may be either:

(a) A simple regression component, $z_t = \omega x_t$ (here x_t is called a simple input) or

(b) A transfer function model component which allows for the effect of lagged values of the variable, related to x_t by

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

The noise n_t is assumed to follow a (possibly seasonal) ARIMA model, i.e., may be represented in terms of an uncorrelated series, a_t, by the hierarchy of equations:

$$\begin{array}{ll}\nabla^d \nabla_s^D n_t & c + w_t \\ w_t & \Phi_1 w_{t-s} + \Phi_2 w_{t-2\times s} + \cdots + \Phi_P w_{t-P\times s} + e_t - \Theta_1 e_{t-s} - \Theta_2 e_{t-2\times s} - \cdots - \Theta_Q e_{t-Q\times s} \\ e_t & \phi_1 e_{t-1} + \phi_2 e_{t-2} + \cdots + \phi_p e_{t-p} + a_t - \theta_1 a_{t-1} - \theta_2 a_{t-2} - \cdots - \theta_q a_{t-q}\end{array}$$

Note: the orders p, q appearing in each of the transfer function models and the ARIMA model are not necessarily the same; $\nabla^d \nabla_s^D n_t$ is the result of applying non-seasonal differencing of order d and seasonal differencing of seasonality s and order D to the series n_t, the differenced series is then of length $N = n - d - s \times D$; the constant term argument c may optionally be held fixed at its initial value (usually, but not necessarily zero) rather than being estimated.

For the purpose of defining an estimation criterion it is assumed that the series a_t is a sequence of independent Normal variates having mean 0 and variance σ_a^2. An allowance has to be made for the effects of unobserved data prior to the observation period. For the noise component an allowance is always made using a form of backforecasting.

For each transfer function input, you have to decide what values are to be assumed for the pre-period terms $z_0, z_{-1}, \ldots, z_{1-p}$ and $x_0, x_{-1}, \ldots, x_{1-b-q}$ which are in theory necessary to re-create the component series $z_1, z_2, \ldots, z_n$, during the estimation procedure.

The first choice is to assume that all these values are zero. In this case in order to avoid undesirable transient distortion of the early values $z_1, z_2, \ldots,$ you are advised first to correct the input series x_t by subtracting from all the terms a suitable constant to make the early values $x_1, x_2, \ldots,$ close to zero. The series mean $\bar{x}$ is one possibility, but for a series with strong trend, the constant might be simply x_1.

The second choice is to treat the unknown pre-period terms as nuisance arguments and estimate them along with the other arguments. This choice should be used with caution. For example, if $p = 1$ and $b = q = 0$, it is equivalent to fitting to the data a decaying geometric curve of the form $A\delta^t$, for $t = 1, 2, \ldots$, along with the other inputs, this being the form of the transient. If the output y_t contains a strong trend of this form, which is not otherwise represented in the model, it will have a tendency to influence the estimate of δ away from the value appropriate to the transfer function model.

In most applications the first choice should be adequate, with the option possibly being used as a refinement at the end of the modelling process. The number of nuisance arguments is then $\max(p, b + q)$, with a corresponding loss of degrees of freedom in the residuals. If you align the input x_t with the output by using in its place the shifted series x_{t-b}, then setting $b = 0$ in the transfer function model, there is some improvement in efficiency. On some occasions when the model contains two or more inputs, each with estimation of pre-period nuisance arguments, these arguments may be co-linear and lead to failure of the function. The option must then be 'switched off' for one or more inputs.

3.2 The Estimation Criterion

This is a measure of how well a proposed set of arguments in the transfer function and noise ARIMA models, matches the data. The estimation function searches for argument values which minimize this criterion. For a proposed set of argument values it is derived by calculating:

(i) the components $z_{1,t}, z_{2,t}, \ldots, z_{m,t}$ as the responses to the input series $x_{1,t}, x_{2,t}, \ldots, x_{m,t}$ using the equations (a) or (b) above,

(ii) the discrepancy between the output and the sum of these components, as the noise

$$n_t = y_t - \left(z_{1,t} + z_{2,t} + \cdots + z_{m,t}\right),$$

(iii) the residual series a_t from n_t by reversing the recursive equations (c), (d) and (e) above.

This last step again requires treatment of the effect of unknown pre-period values of n_t and other terms in the equations regenerating a_t. One approach is to use a sum of squares function as the estimation criteria, which is equivalent to taking the infinite set of past values $n_0, n_{-1}, n_{-2}, \ldots,$ as (linear) nuisance arguments. There is no loss of degrees of freedom however, because the sum of squares function S may be expressed as including the corresponding set of past residuals – see Box and Jenkins (1976) page 273, who prove that

$$S = \sum_{-\infty}^{n} a_t^2.$$

The function $D = S$ is the first of the three possible criteria, and is quite adequate for moderate to long series with no seasonal arguments. The second is the exact likelihood criterion which considers the past set $n_0, n_{-1}, n_{-2}, \ldots$, not as simple nuisance arguments, but as unobserved random variables with known distribution. Calculation of the likelihood of the observed set $n_1, n_2, \ldots, n_n$ requires theoretical integration over the range of the past set. Fortunately this yields a criterion of the form $D = M \times S$ (whose minimization is equivalent to maximizing the exact likelihood of the data), where S is exactly as before, and the multiplier M is a function calculated from the ARIMA model arguments. The value of M is always ≥ 1, and M tends to 1 for any fixed argument set as the sample size n tends to ∞. There is a moderate computational overhead in using this option, but its use avoids appreciable bias in the ARIMA model arguments and yields a better conditioned estimation problem.

The third criterion of marginal likelihood treats the coefficients of the simple inputs in a manner analogous to that given to the past set $n_0, n_{-1}, n_{-2}, \ldots$. These coefficients, together with the constant term c used to represent the mean of w_t, are in effect treated as random variables with highly dispersed distributions. This leads to the criterion $D = M \times S$ again, but with a different value of M which now depends on the simple input series values x_t. In the presence of a moderate to large number of simple inputs, the marginal likelihood criterion can counteract bias in the ARIMA model arguments which is caused by estimation of the simple inputs. This is particularly important in relatively short series.

nag_tsa_multi_inp_model_estim (g13bec) can be used with no input series present, to estimate a univariate ARIMA model for the ouput alone. The marginal likelihood criterion is then distinct from exact likelihood only if a constant term is being estimated in the model, because this is treated as an implicit simple input.

3.3 The Estimation Procedure

This is the minimization of the estimation criterion or objective function D (for deviance). The function uses an extension of the algorithm of Marquardt (1963). The step size in the minimization is inversely related to an argument α, which is increased or decreased by a factor β at successive iterations, depending on the progress of the minimization. Convergence is deemed to have occurred if the fractional reduction of D in successive iterations is less than a value γ, while $\alpha < 1$.

Certain model arguments (in fact all excluding the ω's) are subject to stability constraints which are checked throughout to within a specified tolerance multiple δ of machine accuracy. Using the least squares criterion, the minimization may halt prematurely when some arguments 'stick' at a constraint boundary. This can happen particularly with short seasonal series (with a small number of whole seasons). It will not happen using the exact likelihood criterion, although convergence to a point on the boundary may sometimes be rather slow, because the criterion function may be very flat in such a region. There is also a smaller risk of a premature halt at a constraint boundary when marginal likelihood is used.

A positive, or zero number of iterations can be specified. In either case, the value D of the objective function at iteration zero is computed at the initial argument values, except for the estimation of any pre-period terms for the input series, backforecasts for the noise series, and the coefficients of any simple inputs, and the constant term (unless this is held fixed).

At any later iteration, the value of D is computed after re-estimation of the backforecasts to their optimal values, corresponding to the model arguments presented at that iteration. This is not true for any pre-period terms for the input series which, although they are updated from the previous iteration, may not be precisely optimal for the argument values presented, unless convergence of those arguments has occurred. However, in the case of marginal likelihood being specified, the coefficients of the simple inputs and the constant term are also re-estimated together with the backforecasts at each iteration, to values which are optimal for the other argument values presented.

3.4 Further Results

The residual variance is taken as $erv = \frac{S}{df}$ where $df = N-$ (total number of arguments estimated), is the residual degrees of freedom (for definition of S see Section 3.2 and for definition of N see Section 3.1). The pre-period nuisance arguments for the input series are included in the reduction of df, as is the constant if it is estimated.

The covariance matrix of the vector of model parameter estimates is given by

$$erv \times H^{-1}$$

where H is the linearised least squares matrix taken from the final iteration of the algorithm of Marquardt. From this expression are derived the vector of standard deviations, and the correlation matrix of parameter estimates. These are approximations which are only valid asymptotically, and must be treated with great caution when the parameter estimates are close to their constraint boundaries.

The residual series a_t is available upon completion of the iterations over the range $t = 1 + d + s \times D, \ldots, n$ corresponding to the differenced noise series w_t.

Because of the algorithm used for backforecasting, these are only true residuals for $t \geq 1 + q + s \times Q - p - s \times P - d - s \times D$, provided this is positive. Estimation of pre-period terms for the inputs will also tend to reduce the magnitude of the early residuals, sometimes severely.

The model component series $z_{1,t}, \ldots, z_{m,t}$ and n_t may optionally be returned in order to assess the effects of the various inputs on the output.

4 References

Box G E P and Jenkins G M (1976) *Time Series Analysis: Forecasting and Control* (Revised Edition) Holden–Day

Byng M Fitting a seasonal ARIMA model using the NAG C Library *NAG Technical Report* **TR3/09** http://www.nag.co.uk/technical-report-repository#tr0309

Marquardt D W (1963) An algorithm for least squares estimation of nonlinear parameters *J. Soc. Indust. Appl. Math.* **11** 431

5 Arguments

1: **arimav** – Nag_ArimaOrder *

Pointer to structure of type Nag_ArimaOrder with the following members:

p – Integer
d – Integer *Input*
q – Integer *Input*
bigp – Integer *Input*
bigd – Integer *Input*
bigq – Integer *Input*
s – Integer *Input*

On entry: these seven members of **arimav** must specify the orders vector (p, d, q, P, D, Q, s), respectively, of the ARIMA model for the output noise component.

p, q, P and Q refer, respectively, to the number of autoregressive (ϕ), moving average (θ), seasonal autoregressive (Φ) and seasonal moving average (Θ) arguments.

d, D and s refer, respectively, to the order of non-seasonal differencing, the order of seasonal differencing and the seasonal period.

Constraints:

$p,\ d,\ q,\ P,\ D,\ Q,\ s \geq 0$;
$p + q + P + Q > 0$;
$s \neq 1$;
if $s = 0,\ P + D + Q = 0$;
if $s > 1,\ P + D + Q > 0$;
$d + s \times (P + D) \leq n$;
$p + d - q + s \times (P + D - Q) \leq n$.

2: **nseries** – Integer *Input*

On entry: the total number of input and output series. There may be any number of input series (including none), but always one output series.

Constraint: **nseries** > 1 if there are no arguments in the model (that is $p = q = P = Q = 0$ and **options.cfixed** = Nag_TRUE), **nseries** ≥ 1 otherwise.

3: **transfv** – Nag_TransfOrder *

Pointer to structure of type Nag_TransfOrder with the following members:

b – Integer * *Input/Output*
q – Integer * *Input/Output*
p – Integer *
r – Integer * *Input/Output*

On entry/exit: before use these member pointers **must** be allocated memory by calling nag_tsa_transf_orders (g13byc) which allocates **nseries** $-$ 1 elements to each pointer. The memory allocated to these pointers must be given the transfer function model orders b, q

and p of each of the input series. The order arguments for input series i are held in the ith element of the allocated memory for each pointer. **transfv→b**$[i-1]$ holds the value b_i, **transfv→q**$[i-1]$ holds the value q_i and **transfv→p**$[i-1]$ holds the value p_i. For a simple input, $b_i = q_i = p_i = 0$. **transfv→r**$[i-1]$ holds the value r_i, where $r_i = 1$ for a simple input, $r_i = 2$ for a transfer function input for which no allowance is to be made for pre-observation period effects, and $r_i = 3$ for a transfer function input for which pre-observation period effects will be treated by estimation of appropriate nuisance arguments. When $r_i = 1$, any nonzero contents of the ith element of **transfv→b**, **transfv→q** and **transfv→p** are ignored.

Constraint: **transfv→r**$[i-1] = 1$, 2 or 3, for $i = 1, 2, \dots,$ **nseries** $- 1$

The memory allocated to the members of transfv must be freed by a call to nag_tsa_trans_free (g13bzc)

4: **para**[**npara**] – double *Input/Output*

On entry: initial values of the multi-input model arguments. These are in order, firstly the ARIMA model arguments: p values of ϕ arguments, q values of θ arguments, P values of Φ arguments and Q values of Θ arguments. These are followed by initial values of the transfer function model arguments $\omega_0, \omega_1, \dots, \omega_{q_1}$, $\delta_1, \delta_2, \dots, \delta_{p_1}$ for the first of any input series and similarly for each subsequent input series. The final component of **para** is the initial value of the constant c, whether it is fixed or is to be estimated.

On exit: the latest values of the estimates of these arguments.

5: **npara** – Integer *Input*

On entry: the exact number of ϕ, θ, Φ, Θ, ω, δ and c arguments.

Constraint: **npara** $= p + q + P + Q +$ **nseries** $+ \sum(p_i + q_i)$, the summation being over all the input series. (c must be included, whether fixed or estimated.)

6: **nxxy** – Integer *Input*

On entry: the (common) length of the original, undifferenced input and output time series.

7: **xxy**[**nxxy** $\times$ **tdxxy**] – const double *Input*

Note: the (i, j)th element of the matrix is stored in **xxy**$[(i-1) \times$ **tdxxy** $+ j - 1]$.

On entry: the columns of **xxy** must contain the **nxxy** original, undifferenced values of each of the input series, x_t, and the output series, y_t, in that order.

8: **tdxxy** – Integer *Input*

On entry: the stride separating matrix column elements in the array **xxy**.

Constraint: **tdxxy** $\geq$ **nseries**.

9: **sd**[**npara**] – double *Output*

On exit: the **npara** values of the standard deviations corresponding to each of the arguments in **para**. When the constant is fixed its standard deviation is returned as zero. When the values of **para** are valid, the values of **sd** are usually also valid unless the function fails to invert the second derivative matrix in which case **fail** will have an exit value of NE_MAT_NOT_POS_-DEF.

10: **rss** – double * *Output*

On exit: the residual sum of squares, S, at the latest set of valid parameter estimates.

11: **objf** – double * *Output*

On exit: the objective function, D, at the latest set of valid parameter estimates.

12: **df** – double * *Output*

On exit: the degrees of freedom associated with S.

13: **options** – Nag_G13_Opt * *Input/Output*

On entry/exit: a pointer to a structure of type Nag_G13_Opt whose members are optional parameters for nag_tsa_multi_inp_model_estim (g13bec). If the optional parameters are not required, then the null pointer, `G13_DEFAULT`, can be used in the function call to nag_tsa_multi_inp_model_estim (g13bec). Details of the optional parameters and their types are given below in Section 11.2.

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

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

6 Error Indicators and Warnings

If the intermediate results of optimization are written to a file using the optional parameter **options.outfile**, then NE_NOT_APPEND_FILE, NE_WRITE_ERROR and NE_NOT_CLOSE_FILE could also occur.

NE_2_INT_ARG_LT

On entry, **tdxxy** $= \langle value \rangle$ while **nseries** $= \langle value \rangle$. These arguments must satisfy **tdxxy** $\geq$ **nseries**.

NE_ALLOC_FAIL

Dynamic memory allocation failed.

NE_ARIMA_TEST_FAILED

On entry, or during execution, one or more sets of the ARIMA (ϕ, θ, Φ or Θ) arguments do not satisfy the stationarity or invertibility test conditions.

NE_BAD_PARAM

On entry, argument **options.cfixed** had an illegal value.

On entry, argument **options.criteria** had an illegal value.

On entry, argument **options.print_level** had an illegal value.

NE_CONSTRAINT

General constraint: $\langle value \rangle$.

NE_DELTA_TEST_FAILED

On entry, or during execution, one or more sets of δ arguments do not satisfy the stationarity or invertibility test conditions.

NE_DIFORDER_LEN_INCONSIST

The orders of differencing specified in the structure arimav must satisfy **nxxy** $>$ **arimav→d**$+$(**arimav→s** $*$ **arimav→bigd**), **nxxy** $= \langle value \rangle$, **arimav→d** $= \langle value \rangle$, **arimav→s** $= \langle value \rangle$, **arimav→bigd** $= \langle value \rangle$.

NE_G13_OPTIONS_NOT_INIT

On entry, the option structure, **options**, has not been initialized using nag_tsa_options_init (g13bxc).

NE_G13_ORDERS_NOT_INIT

On entry, the orders array structure, **transfv**, has not been successfully initialized using function nag_tsa_transf_orders (g13byc).

NE_INT_ARG_LT

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

On entry, **options**.**gamma** = ⟨*value*⟩.
Constraint: **options**.**gamma** ≥ 0.0.

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

NE_INT_ARRAY_2

Value ⟨*value*⟩ given to **transfv**.**transfv→r**[⟨*value*⟩] not valid. Correct range for elements of **transfv**.**transfv→r** is $1 \leq$ **transfv→r**$[i] \leq 3$.

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.

NE_INVALID_NSER

On entry, **nseries** = 1 and there are no arguments in the model, i.e., ($p = q = P = Q = 0$ and **options**.**cfixed** = Nag_TRUE).

NE_ITER_FAIL_NIT

The function has failed to converge after **options**.**max_iter** iterations, where **options**.**max_iter** = ⟨*value*⟩. If steady decreases in the objective function, D, were monitored up to the point where this exit occurred, see the optional parameter **options**.**print_level**, then **options**.**max_iter** was probably set too small. If so the calculations should be restarted from the final point held in **para**.

NE_MAT_NOT_POS_DEF

Attempt to invert the second derivative matrix needed in the calculation of the covariance matrix of the parameter estimates has failed. The matrix is not positive definite, possibly due to rounding errors.

NE_NOT_APPEND_FILE

Cannot open file ⟨*string*⟩ for appending.

NE_NOT_CLOSE_FILE

Cannot close file ⟨*string*⟩.

NE_NPARA_MR_MT_INCONSIST

On entry, there is inconsistency between **npara** on the one hand and the elements in the orders structures, **arimav** and **transfv** on the other.

NE_NSER_INCONSIST

Value of **nseries** passed to nag_tsa_transf_orders (g13byc) was ⟨*value*⟩ which is not equal to the value ⟨*value*⟩ passed in this function.

NE_REAL_ARG_GE

On entry, **options**.**gamma** must not be greater than or equal to 1.0: **options**.**gamma** = ⟨*value*⟩.

NE_REAL_ARG_LE

On entry, **options.alpha** $= \langle value \rangle$.
Constraint: **options.alpha** > 0.0.

On entry, **options.beta** $= \langle value \rangle$.
Constraint: **options.beta** > 1.0.

NE_REAL_ARG_LT

On entry, **options.delta** $= \langle value \rangle$.
Constraint: **options.delta** ≥ 1.0.

NE_SOLUTION_FAIL_CONV

Iterative refinement has failed to improve the solution of the equations giving the latest estimates of the arguments. This occurred because the matrix of the set of equations is too ill-conditioned.

NE_WRITE_ERROR

Error occurred when writing to file $\langle string \rangle$.

7 Accuracy

The computation used is believed to be stable.

8 Parallelism and Performance

nag_tsa_multi_inp_model_estim (g13bec) is not threaded in any implementation.

9 Further Comments

The time taken by nag_tsa_multi_inp_model_estim (g13bec) is approximately proportional to **nxxy** $\times$ **options.iter** $\times$ **npara**^{2}.

10 Example

The data in the example relate to 40 observations of an output time series and of a single input time series. The noise series has one autoregressive (ϕ) and one seasonal moving average (Θ) argument (both of which are initially set to zero) for which the seasonal period is 4. The input series is defined by orders $b_1 = 1$, $q_1 = 0$, $p_1 = 1$, $r_1 = 3$, so that it has one ω (initially set to 2.0) and one δ (initially set to 0.5), and allows for pre-observation period effects. The constant (initially set to zero) is to be estimated so that the flag for the constant c, **options.cfixed**, remains unchanged from its default value of Nag_FALSE. Default values of **zsp** are used. Up to 20 iterations are allowed so that **options.max_iter** is set to 20, and the progress of these is monitored and solution output by setting **options.print_level** = Nag_Soln_Iter_Full. Marginal likelihood is the chosen estimation criterion so that **options.criteria** = Nag_Marginal.

After the successful call to nag_tsa_multi_inp_model_estim (g13bec), the following are computed and printed out: the correlation matrix, the residuals for the 36 differenced values and the values of z_t and n_t.

An additional example showing how to use nag_tsa_multi_inp_model_estim (g13bec) to fit a seasonal ARIMA model can be found in Byng.

10.1 Program Text

```
/* nag_tsa_multi_inp_model_estim (g13bec) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

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

#define XXY(I, J) xxy[(I) *tdxxy + J]

int main(void)
{
  Integer exit_status = 0;
  Integer i, inser, j, npara, nseries, nxxy, tdxxy;
  Nag_ArimaOrder arimav;
  Nag_G13_Opt options;
  Nag_TransfOrder transfv;
  double df, objf, *para = 0, rss, *sd = 0, *xxy = 0;
  NagError fail;

  INIT_FAIL(fail);

  printf("nag_tsa_multi_inp_model_estim (g13bec) Example Program Results\n");
#ifdef _WIN32
  scanf_s(" %*[^\n]"); /* Skip heading in data file */
#else
  scanf(" %*[^\n]"); /* Skip heading in data file */
#endif

#define CM(I, J) options.cm[(J)+(I) *options.tdcm]
#define ZT(I, J) options.zt[(J)+(I) *options.tdzt]

  /*
   * Initialize the option structure.
   */
  /* nag_tsa_options_init (g13bxc).
   * Initialization function for option setting
   */
  nag_tsa_options_init(&options);

#ifdef _WIN32
  scanf_s("%" NAG_IFMT "%" NAG_IFMT "%" NAG_IFMT "", &nxxy, &nseries,
          &options.max_iter);
#else
  scanf("%" NAG_IFMT "%" NAG_IFMT "%" NAG_IFMT "", &nxxy, &nseries,
        &options.max_iter);
#endif
  if (nxxy > 0 && nseries > 0) {
    /*
     * Set some specific option variables to the desired values.
     */

    options.criteria = Nag_Marginal;
    options.print_level = Nag_Soln_Iter_Full;
    /*
     * Allocate memory to the arrays in structure transfv containing
     * the transfer function model orders of the input series.
     */
    /* nag_tsa_transf_orders (g13byc), see above. */
    nag_tsa_transf_orders(nseries, &transfv, &fail);
```

```
    /*
     * Read the orders vector of the ARIMA model for the output noise
     * component into structure arimav.
     */
#ifdef _WIN32
    scanf_s("%" NAG_IFMT "%" NAG_IFMT "%" NAG_IFMT "%" NAG_IFMT "%" NAG_IFMT
            "" "%" NAG_IFMT "%" NAG_IFMT "", &arimav.p, &arimav.d, &arimav.q,
            &arimav.bigp, &arimav.bigd, &arimav.bigq, &arimav.s);
#else
    scanf("%" NAG_IFMT "%" NAG_IFMT "%" NAG_IFMT "%" NAG_IFMT "%" NAG_IFMT ""
          "%" NAG_IFMT "%" NAG_IFMT "", &arimav.p, &arimav.d, &arimav.q,
          &arimav.bigp, &arimav.bigd, &arimav.bigq, &arimav.s);
#endif
    /*
     * Read the transfer function model orders of the input series into
     * structure transfv.
     */
    inser = nseries - 1;

    for (j = 0; j < inser; ++j)
#ifdef _WIN32
      scanf_s("%" NAG_IFMT "", &transfv.b[j]);
#else
      scanf("%" NAG_IFMT "", &transfv.b[j]);
#endif
    for (j = 0; j < inser; ++j)
#ifdef _WIN32
      scanf_s("%" NAG_IFMT "", &transfv.q[j]);
#else
      scanf("%" NAG_IFMT "", &transfv.q[j]);
#endif
    for (j = 0; j < inser; ++j)
#ifdef _WIN32
      scanf_s("%" NAG_IFMT "", &transfv.p[j]);
#else
      scanf("%" NAG_IFMT "", &transfv.p[j]);
#endif
    for (j = 0; j < inser; ++j)
#ifdef _WIN32
      scanf_s("%" NAG_IFMT "", &transfv.r[j]);
#else
      scanf("%" NAG_IFMT "", &transfv.r[j]);
#endif

    npara = 0;
    for (i = 0; i < inser; ++i)
      npara = npara + transfv.q[i] + transfv.p[i];
    npara = npara + arimav.p + arimav.q + arimav.bigp + arimav.bigq + nseries;
    if (npara >= 1) {
      if (!(para = NAG_ALLOC(npara, double)) ||
          !(sd = NAG_ALLOC(npara, double)) ||
          !(xxy = NAG_ALLOC(nxxy * nseries, double)))
      {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
      }
      tdxxy = nseries;
      for (i = 0; i < npara; ++i)
#ifdef _WIN32
        scanf_s("%lf", &para[i]);
#else
        scanf("%lf", &para[i]);
#endif
      for (i = 0; i < nxxy; ++i)
        for (j = 0; j < nseries; ++j)
#ifdef _WIN32
          scanf_s("%lf", &XXY(i, j));
#else
          scanf("%lf", &XXY(i, j));
#endif
```

```
      /* nag_tsa_multi_inp_model_estim (g13bec), see above. */
      fflush(stdout);
      nag_tsa_multi_inp_model_estim(&arimav, nseries, &transfv, para,
                                    npara, nxxy, xxy, tdxxy, sd, &rss,
                                    &objf, &df, &options, &fail);
      if (fail.code != NE_NOERROR) {
        printf("Error from nag_tsa_multi_inp_model_estim (g13bec)"
               ".\n%s\n", fail.message);
        exit_status = 1;
        goto END;
      }

      printf("\nThe correlation matrix is \n\n");
      for (i = 0; i < npara; ++i)
        for (j = 0; j < npara; ++j)
          printf("%10.4f%c", CM(i, j), (j % 5 == 4) ? '\n' : ' ');
      printf("\nThe residuals and the z and n values are\n\n");
      printf("   i        res[i]          z(t)        noise(t)\n\n");
      for (i = 0; i < nxxy; ++i) {
        if (i + 1 <= options.lenres) {
          printf("%4" NAG_IFMT "%15.3f", i + 1, options.res[i]);
          for (j = 0; j < nseries - 1; ++j)
            printf("%15.3f ", ZT(i, j));
          printf("%15.3f\n", options.noise[i]);
        }
      }
    }
    else {
      printf("npara is out of range: npara = %-3" NAG_IFMT "\n", npara);
      /* nag_tsa_free (g13xzc).
       * Freeing function for use with g13 option setting
       */
      nag_tsa_free(&options);
      /* nag_tsa_trans_free (g13bzc), see above. */
      nag_tsa_trans_free(&transfv);
      exit_status = 1;
      goto END;
    }
  }
  else {
    printf("One or both of nxxy and nseries are out of range:"
           " nxxy = %-3" NAG_IFMT "  while  nseries = %-3" NAG_IFMT "\n",
           nxxy, nseries);
    exit_status = 1;
    goto END;
  }
  /* nag_tsa_trans_free (g13bzc), see above. */
  nag_tsa_trans_free(&transfv);
  /* nag_tsa_free (g13xzc), see above. */
  nag_tsa_free(&options);

END:
  NAG_FREE(para);
  NAG_FREE(sd);
  NAG_FREE(xxy);

  return exit_status;
}
```

10.2 Program Data

```
nag_tsa_multi_inp_model_estim (g13bec) Example Program Data
 40    2    20
    1    0    0    0    0    1    4
    1
    0
    1
    3
0.0       0.0       2.0       0.5       0.0
```

```
     8.075     105.0
     7.819     119.0
     7.366     119.0
     8.113     109.0
     7.380     117.0
     7.134     135.0
     7.222     126.0
     7.768     112.0
     7.386     116.0
     6.965     122.0
     6.478     115.0
     8.105     115.0
     8.060     122.0
     7.684     138.0
     7.580     135.0
     7.093     125.0
     6.129     115.0
     6.026     108.0
     6.679     100.0
     7.414      96.0
     7.112     107.0
     7.762     115.0
     7.645     123.0
     8.639     122.0
     7.667     128.0
     8.080     136.0
     6.678     140.0
     6.739     122.0
     5.569     102.0
     5.049     103.0
     5.642      89.0
     6.808      77.0
     6.636      89.0
     8.241      94.0
     7.968     104.0
     8.044     108.0
     7.791     119.0
     7.024     126.0
     6.102     119.0
     6.053     103.0
```

10.3 Program Results

```
nag_tsa_multi_inp_model_estim (g13bec) Example Program Results

Parameters to g13bec
____________________

nseries.....................   2

criteria............ Nag_Marginal    cfixed................. Nag_FALSE
alpha..................  1.00e-02    beta...................  1.00e+01
delta..................  1.00e+03    gamma..................  1.00e-07
print_level... Nag_Soln_Iter_Full
outfile................    stdout

Iter =  -1     Residual =   6.456655e+03     Objf =   7.097184e+03

phi                     0.000000e+00
stheta                  0.000000e+00
omega      series   1   2.000000e+00
delta      series   1   5.000000e-01
constant                8.688399e+01

Iter =   0     Residual =   5.802775e+03     Objf =   6.378435e+03

phi                     0.000000e+00
stheta                  0.000000e+00
omega      series   1   2.000000e+00
delta      series   1   5.000000e-01
```

```
constant                  8.573272e+01

Iter =   1     Residual =   2.354664e+03     Objf =   2.498647e+03

phi                       6.589153e-01
stheta                    6.571389e-02
omega      series   1   3.721182e+00
delta      series   1   5.237968e-01
constant                  5.739128e+01

Iter =   2     Residual =   1.922339e+03     Objf =   2.032375e+03

phi                       6.417690e-01
stheta                   -2.361191e-01
omega      series   1   4.523132e+00
delta      series   1   5.742824e-01
constant                  3.814856e+01

Iter =   3     Residual =   1.530797e+03     Objf =   1.630603e+03

phi                       5.550797e-01
stheta                   -3.097333e-01
omega      series   1   7.697297e+00
delta      series   1   7.358370e-01
constant                 -9.322197e+01

Iter =   4     Residual =   1.232926e+03     Objf =   1.324116e+03

phi                       3.698329e-01
stheta                   -2.145294e-01
omega      series   1   9.116523e+00
delta      series   1   6.923742e-01
constant                 -9.985550e+01

Iter =   5     Residual =   1.200813e+03     Objf =   1.289272e+03

phi                       3.889281e-01
stheta                   -2.649652e-01
omega      series   1   8.906746e+00
delta      series   1   6.659905e-01
constant                 -7.782515e+01

Iter =   6     Residual =   1.197922e+03     Objf =   1.286734e+03

phi                       3.752731e-01
stheta                   -2.499956e-01
omega      series   1   8.957172e+00
delta      series   1   6.616140e-01
constant                 -7.656262e+01

Iter =   7     Residual =   1.197934e+03     Objf =   1.286623e+03

phi                       3.804046e-01
stheta                   -2.594526e-01
omega      series   1   8.954182e+00
delta      series   1   6.599012e-01
constant                 -7.553429e+01

Iter =   8     Residual =   1.198009e+03     Objf =   1.286613e+03

phi                       3.807082e-01
stheta                   -2.567453e-01
omega      series   1   8.956063e+00
delta      series   1   6.597438e-01
constant                 -7.549190e+01

Iter =   9     Residual =   1.197988e+03     Objf =   1.286612e+03

phi                       3.808772e-01
stheta                   -2.580559e-01
omega      series   1   8.955983e+00
```

```
delta      series   1   6.596508e-01
constant               -7.543851e+01

Iter =  10     Residual =   1.198002e+03     Objf =   1.286611e+03

phi                     3.809218e-01
stheta                 -2.575832e-01
omega      series   1   8.956106e+00
delta      series   1   6.596484e-01
constant               -7.544005e+01

Iter =  11     Residual =   1.197997e+03     Objf =   1.286611e+03

phi                     3.809235e-01
stheta                 -2.577863e-01
omega      series   1   8.956084e+00
delta      series   1   6.596411e-01
constant               -7.543552e+01



The number of iterations carried out is   11

The final values of the parameters and their standard deviations are

   i            para[i]                sd
   1             0.380924            0.166379
   2            -0.257786            0.178178
   3             8.956084            0.948061
   4             0.659641            0.060239
   5           -75.435521           33.505341


The residual sum of squares =     1.197997e+03

The objective function =     1.286611e+03

The degrees of freedom =    34.00

The correlation matrix is

    1.0000    -0.1839    -0.1775    -0.0340     0.1394
   -0.1839     1.0000     0.0518     0.2547    -0.2860
   -0.1775     0.0518     1.0000    -0.3070    -0.2926
   -0.0340     0.2547    -0.3070     1.0000    -0.8185
    0.1394    -0.2860    -0.2926    -0.8185     1.0000

The residuals and the z and n values are

   i         res[i]          z(t)        noise(t)

   1          0.397        180.567         -75.567
   2          3.086        191.430         -72.430
   3         -2.818        196.302         -77.302
   4         -9.941        195.460         -86.460
   5         -5.061        201.594         -84.594
   6         14.053        199.076         -64.076
   7          2.624        195.211         -69.211
   8         -5.823        193.450         -81.450
   9         -2.147        197.179         -81.179
  10         -0.216        196.217         -74.217
  11         -2.517        191.812         -76.812
  12          7.916        184.544         -69.544
  13          1.423        194.322         -72.322
  14         11.936        200.369         -62.369
  15          5.117        200.990         -65.990
  16         -5.672        200.468         -75.468
  17         -5.681        195.763         -80.763
  18         -1.637        184.025         -76.025
  19         -1.019        175.360         -75.360
  20         -2.623        175.492         -79.492
```

```
 21          3.283        182.162          -75.162
 22          6.896        183.857          -68.857
 23          5.395        190.797          -67.797
 24          0.875        194.327          -72.327
 25         -4.153        205.558          -77.558
 26          6.206        204.261          -68.261
 27          4.208        207.104          -67.104
 28         -2.387        196.423          -74.423
 29        -11.803        189.924          -87.924
 30          6.435        175.158          -72.158
 31          1.342        160.761          -71.761
 32         -4.924        156.575          -79.575
 33          4.799        164.256          -75.256
 34         -0.074        167.783          -73.783
 35         -6.023        184.483          -80.483
 36         -6.427        193.055          -85.055
 37         -2.527        199.390          -80.390
 38          2.039        201.302          -75.302
 39          0.243        195.695          -76.695
 40         -3.166        183.738          -80.738
```

11 Optional Parameters

A number of optional input and output arguments to nag_tsa_multi_inp_model_estim (g13bec) are available through the structure argument **options** of type Nag_G13_Opt. a argument may be selected by assigning an appropriate value to the relevant structure member. Those arguments not selected will be assigned default values. If no use is to be made of any of the optional parameters you should use the null pointer, `G13_DEFAULT`, in place of **options** when calling nag_tsa_multi_inp_model_estim (g13bec); the default settings will then be used for all arguments.

Before assigning values to **options** the structure must be initialized by a call to the function nag_tsa_options_init (g13bxc). Values may then be assigned directly to the structure members in the normal C manner.

Options selected are checked within nag_tsa_multi_inp_model_estim (g13bec) for being within the required range, if outside the range, an error message is generated.

When all calls to nag_tsa_multi_inp_model_estim (g13bec) have been completed and the results contained in the options structure are no longer required; then nag_tsa_free (g13xzc) should be called to free the NAG allocated memory from **options**.

11.1 Optional Parameters Checklist and Default Values

For easy reference, the following list shows the input and output members of **options** which are valid for nag_tsa_multi_inp_model_estim (g13bec) together with their default values where relevant. ϵ is the ***machine precision***.

`Boolean` **options.list**	Nag_TRUE
`Nag_PrintType` **options.print_level**	Nag_Soln
`char outfile[80]`	`stdout`
`void (*`**options.print_fun**`)()`	**NULL**
`Boolean` **options.cfixed**	Nag_FALSE
`Nag_Likelihood` **options.criteria**	Nag_Exact
`Integer` **options.max_iter**	50
`double` **options.alpha**	0.01
`double` **options.beta**	10.0
`double` **options.delta**	1000.0
`double` **options.gamma**	$\max(100\epsilon, 10^{-7})$
`Integer` **options.iter**	`double *`**options.cm**
`double *`**options.res**	`Integer` **options.lenres**
`double *`**options.zt**	`double *`**options.noise**

11.2 Description of the Optional Parameters

list – Nag_Boolean

On entry: if **options.list** = Nag_TRUE then the argument settings which are used in the call to nag_tsa_multi_inp_model_estim (g13bec) will be printed.

print_level – Nag_PrintType

On entry: the level of results produced by nag_tsa_multi_inp_model_estim (g13bec). The following values are available:

Nag_NoPrint	No output.
Nag_Soln	The final solution.
Nag_Iter	One line of output for each iteration.
Nag_Soln_Iter	The final solution and one line of output for each iteration.
Nag_Soln_Iter_Full	The final solution and detailed printout at each iteration.

Details of each level of results printout are described in Section 7.3.

Constraint: **options.print_level** = Nag_PrintNotSet, Nag_Soln, Nag_Iter, Nag_Soln_Iter or Nag_Soln_Iter_Full.

outfile – const char[80]

On entry: name of file to which the results of monitoring the course of the optimization should be printed. If **options.outfile**[0] = "\0" then the `stdout` stream is used.

print_fun – pointer to function

On entry: printing function defined by you; the prototype of **options.print_fun** is

```
void (*print_fun)(const Nag_UserPrintFun *bfx, Nag_Comm *Comm);
```

See Section 11.3.1 below for further details.

cfixed – Nag_Boolean

On entry: **options.cfixed** must be set to Nag_TRUE if the constant c is to remain fixed at its initial value, and to Nag_FALSE if it is to be estimated.

criteria – Nag_Likelihood

On entry: indicates the likelihood option for the estimation criterion. **options.criteria** must be set to Nag_LeastSquares, Nag_Exact or Nag_Marginal, to select the least squares, exact or marginal likelihood, respectively.

Constraint: **options.criteria** = Nag_LeastSquares, Nag_Exact or Nag_Marginal.

max_iter – Integer

On entry: the maximum required number of iterations. If **options.max_iter** = 0, no change is made to any of the model arguments in array **para** except that the constant c (if **options.cfixed** = Nag_FALSE) and any ω relating to simple input series are estimated. (Apart from these, estimates are always derived for the nuisance arguments relating to any backforecasts and any pre-observation period effects for transfer function inputs.)

Constraint: **options.max_iter** ≥ 0.

alpha – double

On entry: α, the value used to constrain the magnitude of the search procedure steps (see Section 3.3).

Constraint: **options.alpha** > 0.0.

beta – double

On entry: β, the multiplier which regulates the value of α (see Section 3.3).

Constraint: **options.beta** > 1.0.

delta – double

On entry: δ, the value of the stationarity and invertibility test tolerance factor (see Section 3.3).

Constraint: **options.delta** ≥ 1.0.

gamma – double

On entry: γ, the convergence criterion (see Section 3.3).

Constraint: $0.0 \leq$ **options.gamma** < 1.0.

iter – Integer

On exit: the number of iterations carried out. A value of **options.iter** $= -1$ on exit indicates that the only estimates obtained up to this point have been for the nuisance arguments relating to backforecasts, unless the marginal likelihood option is used in which case estimates have also been obtained for simple input coefficients ω and for the constant c (if **options.cfixed** = Nag_FALSE). This value of **options.iter** usually indicates a failure in a consequent step of estimating transfer function input pre-observation period nuisance arguments. A value of **options.iter** $= 0$ on exit indicates that estimates have been obtained up to this point for the constant c (if **options.cfixed** = Nag_FALSE), for simple input coefficients ω and for the nuisance arguments relating to the backforecasts and to transfer function input pre-observation period effects.

cm – double

On exit: this pointer is allocated memory internally with **npara** $\times$ **npara** elements corresponding to **npara** rows by **npara** columns. The **npara** rows and columns of **options.cm** contain the correlation coefficients relating to each pair of arguments in **para**. All coefficients relating to the constant will be zero if the constant is fixed. However, if the function fails to invert the second derivative matrix, in which case **fail** will have an exit value of NE_MAT_NOT_POS_DEF, then the contents of **options.cm** will be indeterminate.

res – double

On exit: the values of the residuals relating to the differenced values of the output series. This pointer is allocated memory internally with **options.lenres** elements.

lenres – Integer

On exit: the length of **options.res**.

zt – double

On exit: this pointer is allocated memory internally with **nxxy** $\times$ (**nseries** $- 1$) elements corresponding to **nxxy** rows by (**nseries** $- 1$) columns. The columns of **options.zt** hold the values of the input component series z_t.

noise – double

On exit: this pointer is allocated memory internally with **nxxy** elements. It holds the output noise component n_t.

11.3 Description of Printed Output

The level of printed output can be controlled with the structure members **options.list** and **options.print_level**, see section 7.2. If **options.list** = Nag_TRUE then the argument values to nag_tsa_multi_inp_model_estim (g13bec) are listed, whereas the printout of results is governed by the value of **options.print_level**. The default of **options.print_level** = Nag_Soln which provides a printout of the final solution. This section describes all of the possible levels of results printout available from nag_tsa_multi_inp_model_estim (g13bec). When **options.print_level** = Nag_Iter or Nag_Soln_Iter a single line of output is produced at each iteration, this gives the following values.

Iter	the current iteration number, **options.iter**.
Residual	the residual sum of squares, **print_fun→rss**.
Objf	the objective function at the latest set of parameter estimates.

When **options.print_level** = Nag_Soln_Iter_Full a description and value for each of the arguments in the para array is output. The descriptions are phi for ϕ, theta for θ, sphi for Φ. stheta for Θ, omega/si for ω in a simple input, omega for ω in a transfer function input, **options.delta** for δ and constant for c. In addition series 1, series 2, etc, indicate the input series relevant to the omega and **options.delta** arguments.

If **options.print_level** = Nag_Soln, Nag_Soln_Iter or Nag_Soln_Iter_Full the final solution is printed out. This consists of:

i	the argument number.
para[i]	the values of the argument.
sd	the standard deviations.
options.iter	the number of iterations carried out.
rss	the residual sum of squares.
objf	the objective function.
df	the degrees of freedom.

If **options.print_level** = Nag_NoPrint then printout will be suppressed; you can print the final solution when nag_tsa_multi_inp_model_estim (g13bec) returns to the calling program.

11.3.1 Output of results via a user-defined printing function

You may also specify their own print function for output of iteration results and the final solution by use of the **options.print_fun** function pointer, prototype

```
void (*print_fun) (const Nag_UserPrintFun *bfx, Nag_Comm *Comm);
```

The rest of this section can be skipped if the default printing facilities provide the required functionality. When a user-defined function is assigned to **options.print_fun** this will be called in preference to the internal print function of nag_tsa_multi_inp_model_estim (g13bec). Calls to the user-defined function are again controlled by means of the **options.print_level** member. Information is provided through two structure arguments to **options.print_fun**, the structure of type Nag_UserPrintFun contains the following members relevant to nag_tsa_multi_inp_model_estim (g13bec):

itc – Integer

The number of the particular iteration being monitored.

rss – double *

The residual sum of squares, S, at the latest set of valid parameter estimates.

objf – double *

The objective function, D, at the latest set of valid parameter estimates.

para – double *

The pointer to memory containing **print_fun→npara** latest values of the estimates of the multi-input model arguments.

npara – Integer

The exact number of ϕ, θ, Φ, Θ, ω, δ and c arguments.

npe – Integer

The number of ARIMA (ϕ, θ, Φ, Θ), omega (ω), **options.delta** (δ), and c arguments being estimated.

mtyp – Integer
mser – Integer

The pointers to memory, each with **print_fun→npe** elements. The value of each element in **print_fun→mtyp** and **print_fun→mser** corresponds to the description of each parameter estimated in **print_fun→para**. The following should be read in conjunction with the description of the argument **print**. The relevant description for the value of **print_fun→para** is:

print_fun→mtyp[i]	**Description**	
1	phi	
2	theta	
3	sphi	
4	stheta	
5	omega/si	series **print_fun→mser**[i]
6	omega	series **print_fun→mser**[i]
7	**options.delta**	series **print_fun→mser**[i]
8	constant	

for $i = 0, 1, \ldots,$ **print_fun→npe**. For the phi, theta, sphi, stheta and constant arguments, **print_fun→mser**[i] $= 0$.

sd – double *

The pointer to memory containing the npara values of the standard deviations.

df – double *

The number of degrees of freedom associated with S.