NAG Library Function Document

nag_tsa_multi_inp_model_forecast (g13bjc)

1 Purpose

nag_tsa_multi_inp_model_forecast (g13bjc) produces forecasts of a time series (the output series) which may depend on one or more other (input) series via a previously estimated multi-input model. The future values of any input series must be supplied. Standard errors of the forecasts are produced. If future values of some of the input series have been obtained as forecasts using ARIMA models for those series, this may be allowed for in the calculation of the standard errors.

2 Specification

```
#include <nag.h>
#include <nagg13.h>
void nag_tsa_multi_inp_model_forecast (Nag_ArimaOrder *arimav,
     Integer nseries, Nag_TransfOrder *transfv, double para[], Integer npara,
     Integer nev, Integer nfv, const double xxy[], Integer tdxxy,
     double rmsxy[], const Integer mrx[], Integer tdmrx, const double parx[],
     Integer ldparx, Integer tdparx, double fva[], double fsd[],
     Nag_G13_Opt *options, NagError *fail)
```

3 Description

nag_tsa_multi_inp_model_forecast (g13bjc) has two stages. The first stage is essentially the same as a call to the model estimation function nag_tsa_multi_inp_model_estim (g13bec), with zero iterations. In particular, all the arguments remain unchanged in the supplied input series transfer function models and output noise series ARIMA model except that a further iteration takes place for any ω corresponding to a simple input. The internal nuisance arguments associated with the pre-observation period effects of the input series are estimated where requested, and so are any backforecasts of the output noise series. The output components z_t and n_t, and residuals a_t are calculated exactly as described in the Section 3 of nag_tsa_multi_inp_model_estim (g13bec).

In the second stage, the forecasts of the output series y_t are calculated for $t = n+1, n+2, \ldots, n+L$ where n is the latest time point of the observations and L is the maximum lead time of the forecasts.

First the new values, x_t for any input series are used to form the input components z_t for $t = n+1, n+2, \ldots, n+L$ using the transfer function models:

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

The output noise component n_t for $t = n+1, n+2, \ldots, n+L$ is then forecast by setting $a_t = 0$ for $t = n+1, n+2, \ldots, n+L$ and using the ARIMA model equations:

(b) $e_t = \phi_1 e_{t-1} + \phi_2 e_{t-2} + \cdots + \phi_p e_{t-p} + a_t - \theta_1 a_{t-1} - \theta_2 a_{t-2} - \cdots - \theta_q a_{t-q}$.

(c) $w_t = \Phi_1 w_{t-s} + \Phi_2 w_{t-2\times s} + \cdots + \Phi_P w_{t-P\times s} + e_t - \Theta_1 e_{t-s} - \Theta_2 e_{t-2\times s} - \cdots - \Theta_Q e_{t-Q\times s}$.

(d) $n_t = \left(\nabla^d \nabla_s^D\right)^{-1}(w_t + c)$.

This last step of 'integration' reverses the process of differencing. Finally the output forecasts are calculated as

$$y_t = z_{1,t} + z_{2,t} + \cdots + z_{m,t} + n_t.$$

The forecast error variance of y_{t+l} (i.e., at lead time l) is S_l^2, which is the sum of parts which arise from the various input series, and the output noise component. That part due to the output noise is

$$sn_l^2 = V_n \times \left(\psi_0^2 + \psi_1^2 + \cdots + \psi_{l-1}^2\right)$$

V_n is the estimated residual variance of the output noise ARIMA model, and $\psi_0, \psi_1, \ldots$, are the 'psi-weights' of this model as defined in Box and Jenkins (1976). They are calculated by applying the equations (b), (c) and (d) above for $t = 0, 1, \ldots, L$, but with artificial values for the various series and with the constant c set to 0. Thus all values of a_t, e_t, w_t and n_t are taken as zero for $t < 0$; a_t is taken to be 1 for $t = 0$ and 0 for $t > 0$. The resulting values of n_t for $t = 0, 1, \ldots, L$ are precisely $\psi_0, \psi_1, \ldots, \psi_L$ as required.

Further contributions to S_l^2 come only from those input series, for which future values are forecasts which have been obtained by applying input series ARIMA models. For such a series the contribution is

$$sz_l^2 = V_x \times \left(\nu_0^2 + \nu_1^2 + \cdots + \nu_{l-1}^2\right)$$

V_x is the estimated residual variance of the input series ARIMA model. The coefficients $\nu_0, \nu_1, \ldots$ are calculated by applying the transfer function model equation (a) above for $t = 0, 1, \ldots, L$, but again with artificial values of the series. Thus all values of z_t and x_t for $t < 0$ are taken to be zero, and $x_0, x_1, \ldots$ are taken to be the psi-weight sequence $\psi_0, \psi_1, \ldots$ for the input series ARIMA model. The resulting values of z_t for $t = 0, 1, \ldots, L$ are precisely $\nu_0, \nu_1, \ldots, \nu_L$ as required.

In adding such contributions sz_l^2 to sn_l^2 to make up the total forecast error variance S_l^2, it is assumed that the various input series with which these contributions are associated, are statistically independent of each other.

When using the function in practice an ARIMA model is required for all the input series. In the case of those inputs for which no such ARIMA model is available (or its effects are to be excluded), the corresponding orders and arguments and the estimated residual variance should be set to zero.

4 References

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

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 number of input and output series. There may be any number of input series (including none), but only 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*
q – Integer * *Input*
p – Integer *
r – Integer * *Input*

On entry: 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, and $r_i = 2$ or 3 for a transfer function input.

The choice $r_i = 3$ leads to estimation of the pre-period input effects as nuisance arguments, and $r_i = 2$ suppresses this estimation. This choice may affect the returned forecasts.

When $r_i = 1$, any nonzero contents of the ith element of the memory of **transfv**→**b**, **transfv**→**q** and **transfv**→**p** are ignored.

Constraint: **transfv**→**r**$[i-1] = 1$, 2 or 3, for $i = 1, 2, \ldots,$ **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: estimates 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, Q values of Θ arguments. These are followed by the transfer function model argument values $\omega_0, \omega_1, \ldots, \omega_{q_1}$, and $\delta_1, \delta_2, \ldots, \delta_{p_1}$ for the first of any input series and similarly for each subsequent input series. The final component of **para** is the value of the constant c.

On exit: the input estimates are unaltered except that any ω estimates corresponding to a simple input will be undated by a single iteration.

5: **npara** – Integer *Input*

On entry: the exact number of ϕ, θ, Φ, Θ, ω, δ, c arguments, so that **npara** $= p + q + P + Q +$ **nseries** $+ \sum(p_i + q_i)$, the summation being over all the input series. (c must be included whether its value was previously estimated or was set fixed.)

6: **nev** – Integer *Input*

On entry: the number of original (undifferenced) values in each of the input and output time-series.

7: **nfv** – Integer *Input*

On entry: the number of forecast values of the output series required.

Constraint: **nfv** > 0.

8: **xxy**[(**nev** + **nfv**) × **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 in the first **nev** places, the past values of each of the input and output series, in that order. In the next **nfv** places, the columns relating to the input series (i.e., columns 0 to **nseries** $- 2$) contain the future values of the input series which are necessary for construction of the forecasts of the output series y.

9: **tdxxy** – Integer *Input*

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

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

10: **rmsxy**[**nseries**] – double *Input/Output*

On entry: elements of **rmsxy**[0] to **rmsxy**[**nseries** $- 2$] must contain the estimated residual variance of the input series ARIMA models. In the case of those inputs for which no ARIMA model is available or its effects are to be excluded in the calculation of forecast standard errors, the corresponding entry of **rmsxy** should be set to 0.

On exit: **rmsxy**[**nseries** $- 1$] contains the estimated residual variance of the output noise ARIMA model which is calculated from the supplied series. Otherwise **rmsxy** is unchanged.

11: **mrx**[7 × **tdmrx**] – const Integer *Input*

On entry: the orders array for each of the input series ARIMA models. Thus, column $i - 1$ contains values of p, d, q, P, D, Q, s for input series i. In the case of those inputs for which no ARIMA model is available, the corresponding orders should be set to 0.

If there are no input series then **NULL** may be supplied in place of **mrx**.

12: **tdmrx** – Integer *Input*

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

Constraint: **tdmrx** $\geq$ **nseries** $- 1$.

13: **parx**[**ldparx** × **tdparx**] – const double *Input*

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

On entry: values of the arguments (ϕ, θ, Φ, and Θ) for each of the input series ARIMA models. Thus column i contains **mrx**[i] values of ϕ, **mrx**[$(2) \times$ **tdmrx** $+ i$] values of θ, **mrx**[$(3) \times$ **tdmrx** $+ i$] values of Φ and **mrx**[$(5) \times$ **tdmrx** $+ i$] values of Θ – in that order.

Values in the columns relating to those input series for which no ARIMA model is available are ignored.

If there are no input series then **NULL** may be supplied in place of **parx**.

14: **ldparx** – Integer *Input*

On entry: the maximum number of arguments in any of the input series ARIMA models. If there are no input series then **ldparx** is not referenced.

Constraint: **ldparx** $\geq nce = \max(1, (\mathbf{mrx}[i] + \mathbf{mrx}[(2) \times \mathbf{tdmrx} + i] +$
mrx[$(3) \times$ **tdmrx** $+ i$] + **mrx**[$(5) \times$ **tdmrx** $+ i$])), for $i = 0, 1, \ldots,$ **nseries** $- 1$.

15: **tdparx** – Integer *Input*

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

Constraint: **tdparx** $\geq$ **nseries** $-$ 1.

16: **fva**[**nfv**] – double *Output*

On exit: the required forecast values for the output series.

17: **fsd**[**nfv**] – double *Output*

On exit: the standard errors for each of the forecast values.

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

On entry/exit: a pointer to a structure of type Nag_G13_Opt whose members are optional arguments for nag_tsa_multi_inp_model_forecast (g13bjc). If the optional arguments are not required, then the null pointer, `G13_DEFAULT`, can be used in the function call to nag_tsa_multi_inp_model_forecast (g13bjc). Details of the optional arguments and their types are given below in Section 11.

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

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

6 Error Indicators and Warnings

NE_2_INT_ARG_LT

On entry, **ldparx** $= \langle value \rangle$ while $nce = \langle value \rangle$. These arguments must satisfy **ldparx** $\geq nce$. (See the expression for nce in Section 5 where **ldparx** is described).

On entry, **tdmrx** $= \langle value \rangle$ while **nseries** $- 1 = \langle value \rangle$. These arguments must satisfy **tdmrx** $\geq$ **nseries** $- 1$.

On entry, **tdparx** $= \langle value \rangle$ while **nseries** $- 1 = \langle value \rangle$. These arguments must satisfy **tdparx** $\geq$ **nseries** $- 1$.

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.

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_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** in function nag_tsa_transf_orders (g13byc) has not been initialized.

NE_INT_ARG_LE

On entry, **nfv** $= \langle value \rangle$.
Constraint: **nfv** > 0.

NE_INT_ARG_LT

On entry, **nseries** $= \langle value \rangle$.
Constraint: **nseries** ≥ 1.

NE_INT_ARRAY_2

Value $\langle value \rangle$ given to **transfv**$\rightarrow$**r**$[\langle value \rangle]$ not valid. Correct range for elements if **transfv**$\rightarrow$**r** is $1 \leq$ **transfv**$\rightarrow$**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_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_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 $\langle value \rangle$ which is not equal to the value $\langle value \rangle$ passed in this function.

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.

7 Accuracy

The computation used is believed to be stable.

8 Parallelism and Performance

Not applicable.

9 Further Comments

The time taken by nag_tsa_multi_inp_model_forecast (g13bjc) is approximately proportional to the product of the length of each series and the square of the number of arguments in the multi-input model.

10 Example

The data in this example relate to 40 observations of an output time series and 5 input time series. This example differs from Section 10 in nag_tsa_multi_inp_model_estim (g13bec) in nag_tsa_multi_inp_model_estim (g13bec) in that there are now 4 simple input series. The output series has one autoregressive (ϕ) argument and one seasonal moving average (Θ) argument. The seasonal period is 4. The transfer function input (the fifth in the set) is defined by orders $b_5 = 1$, $q_5 = 0$, $p_5 = 1$, $r_5 = 3$, so that it allows for pre-observation period effects. The initial values of the specified model are:

$$\phi = 0.495,\ \Theta = 0.238, \omega_1 = -0.367 \quad \omega_2 = -3.876 \quad \omega_3 = 4.516$$
$$\omega_4 = 2.474 \quad \omega_5 = 8.629 \quad \delta_1 = 0.688,\ c = -82.858.$$

A further eight values of the input series are supplied, and it is assumed that the values for the fifth series have themselves been forecast from an ARIMA model with orders 2 0 2 0 1 1 4, in which $\phi_1 = 1.6743$, $\phi_2 = -0.9505$, $\theta_1 = 1.4605$, $\theta_2 = -0.4862$ and $\Theta_1 = 0.8993$, and for which the residual mean square is 0.1720.

The following are computed and printed out: the estimated residual variance for the output noise series, the eight forecast values and their standard errors, and the values of the components z_t and the output noise component n_t.

10.1 Program Text

```
/* nag_tsa_multi_inp_model_forecast (g13bjc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 2, 1991.
 * Mark 8 revised, 2004.
 */

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

#define PARX(I, J) parx[(I) *tdparx + J]
#define XXY(I, J)  xxy[(I) *tdxxy + J]
#define MRX(I, J)  mrx[(I) *tdmrx + J]

int main(void)
{
  Integer         exit_status = 0;
  Integer         i, inser, j, ldparx, *mrx = 0, n, nev, nfv, npara;
  Integer         nseries, tdmrx, tdparx, tdxxy;
  Nag_ArimaOrder  arimav;
  Nag_G13_Opt     options;
  Nag_TransfOrder transfv;
  double          *fsd = 0, *fva = 0, *para = 0, *parx = 0, *rmsxy = 0;
  double          *xxy = 0;
  NagError        fail;

  INIT_FAIL(fail);

  printf("nag_tsa_multi_inp_model_forecast (g13bjc) Example Program "
         "Results\n");
#ifdef _WIN32
  scanf_s(" %*[^\n]"); /* Skip heading in data file */
#else
  scanf(" %*[^\n]"); /* Skip heading in data file */
```

```
#endif

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

  /*
   * Initialise the option-setting function.
   */
  /* 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"", &nev, &nfv, &nseries);
#else
  scanf("%"NAG_IFMT"%"NAG_IFMT"%"NAG_IFMT"", &nev, &nfv, &nseries);
#endif
  if (nseries > 0 && nev > 0 && nfv > 0)
    {
      /*
       * Set option variable to the desired value.
       */
      options.cfixed = Nag_TRUE;
      /*
       * 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);
      if (fail.code != NE_NOERROR)
        {
          printf("Error from nag_tsa_transf_orders (g13byc).\n%s\n",
                  fail.message);
          exit_status = 1;
          goto END;
        }
      /*
       * 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;
      ldparx = 8;
      if (npara >= 1)
        {
          if (!(fsd = NAG_ALLOC(nfv, double)) ||
              !(fva = NAG_ALLOC(nfv, double)) ||
              !(para = NAG_ALLOC(npara, double)) ||
              !(parx = NAG_ALLOC(ldparx*(nseries-1), double)) ||
              !(rmsxy = NAG_ALLOC(nseries, double)) ||
              !(xxy = NAG_ALLOC((nev+nfv)*(nseries), double)) ||
              !(mrx = NAG_ALLOC(7*(nseries-1), Integer)))
            {
              printf("Allocation failure\n");
              exit_status = -1;
              goto END;
            }
          tdmrx = nseries-1;
          tdparx = nseries-1;
          tdxxy = nseries;

          for (i = 0; i < npara; ++i)
#ifdef _WIN32
            scanf_s("%lf", &para[i]);
#else
            scanf("%lf", &para[i]);
#endif
          n = nev + nfv;
          for (i = 0; i < n; ++i)
            for (j = 0; j < nseries; ++j)
#ifdef _WIN32
              scanf_s("%lf", &XXY(i, j));
#else
              scanf("%lf", &XXY(i, j));
#endif
          for (i = 0; i < nseries; ++i)
#ifdef _WIN32
            scanf_s("%lf", &rmsxy[i]);
#else
            scanf("%lf", &rmsxy[i]);
#endif
          for (i = 0; i < 7; ++i)
            for (j = 0; j < inser; ++j)
#ifdef _WIN32
              scanf_s("%"NAG_IFMT"", &MRX(i, j));
#else
              scanf("%"NAG_IFMT"", &MRX(i, j));
#endif
          for (i = 0; i < 5; ++i)
            for (j = 0; j < inser; ++j)
#ifdef _WIN32
              scanf_s("%lf", &PARX(i, j));
#else
              scanf("%lf", &PARX(i, j));
#endif

          /* nag_tsa_multi_inp_model_forecast (g13bjc), see above. */
          fflush(stdout);
          nag_tsa_multi_inp_model_forecast(&arimav, nseries, &transfv, para,
```

```
                                        npara, nev, nfv, xxy, tdxxy, rmsxy,
                                        mrx, tdmrx, parx, ldparx,
                                        tdparx, fva, fsd, &options, &fail);

          if (fail.code == NE_NOERROR || fail.code == NE_SOLUTION_FAIL_CONV ||
              fail.code == NE_MAT_NOT_POS_DEF)
            {
              printf(
                      "%1"NAG_IFMT" sets of observations were processed.\n",
                      nev);
              printf("\nThe residual mean square for the output ");
              printf("series is %10.4f\n\n", rmsxy[nseries-1]);
              printf(
                      "The forecast values and their standard errors are\n\n");
              printf("\n   i      fva      fsd\n\n");
              for (i = 0; i < nfv; ++i)
                printf("%4"NAG_IFMT"%10.3f%10.4f\n", i+1, fva[i],
                        fsd[i]);
              printf("\nThe values of z(t) and noise(t) are\n\n");
              printf("   i       z1         z2         z3         z4"
                      "         z5       noise\n\n");
              for (i = 0; i < n; ++i)
                {
                  printf("%4"NAG_IFMT"", i+1);
                  for (j = 0; j < nseries-1; ++j)
                    printf("%10.3f ", ZT(i, j));
                  printf("%10.3f\n", options.noise[i]);
                }
            }
          else
            {
              printf(
                      "Error from nag_tsa_multi_inp_model_forecast (g13bjc)."
                      "\n%s\n", fail.message);
              exit_status = 1;
              goto END;
            }
        }
      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 more of nseries, nev and nfv are out of range:"
              " nseries = %-3"NAG_IFMT",  nv = %-3"NAG_IFMT" while  "
              "nfv = %-3"NAG_IFMT"q\n", nseries, nev, nfv);
      exit_status = 1;
      goto END;
    }
  /* nag_tsa_free (g13xzc), see above. */
  nag_tsa_free(&options);
  /* nag_tsa_trans_free (g13bzc), see above. */
  nag_tsa_trans_free(&transfv);

 END:
  NAG_FREE(fsd);
  NAG_FREE(fva);
  NAG_FREE(para);
  NAG_FREE(parx);
  NAG_FREE(rmsxy);
```

```
  NAG_FREE(xxy);
  NAG_FREE(mrx);

  return exit_status;
}
```

10.2 Program Data

```
nag_tsa_multi_inp_model_forecast (g13bjc) Example Program Data
   40    8    6
    1    0    0    0    0    1    4
    0    0    0    0    1
    0    0    0    0    0
    0    0    0    0    1
    1    1    1    1    3
  0.4950  0.2380 -0.3670 -3.8760  4.5160  2.4740  8.6290  0.6880
-82.8580
  1.0     1.0     0.0     0.0     8.075 105.0
  1.0     0.0     1.0     0.0     7.819 119.0
  1.0     0.0     0.0     1.0     7.366 119.0
  1.0    -1.0    -1.0    -1.0     8.113 109.0
  2.0     1.0     0.0     0.0     7.380 117.0
  2.0     0.0     1.0     0.0     7.134 135.0
  2.0     0.0     0.0     1.0     7.222 126.0
  2.0    -1.0    -1.0    -1.0     7.768 112.0
  3.0     1.0     0.0     0.0     7.386 116.0
  3.0     0.0     1.0     0.0     6.965 122.0
  3.0     0.0     0.0     1.0     6.478 115.0
  3.0    -1.0    -1.0    -1.0     8.105 115.0
  4.0     1.0     0.0     0.0     8.060 122.0
  4.0     0.0     1.0     0.0     7.684 138.0
  4.0     0.0     0.0     1.0     7.580 135.0
  4.0    -1.0    -1.0    -1.0     7.093 125.0
  5.0     1.0     0.0     0.0     6.129 115.0
  5.0     0.0     1.0     0.0     6.026 108.0
  5.0     0.0     0.0     1.0     6.679 100.0
  5.0    -1.0    -1.0    -1.0     7.414  96.0
  6.0     1.0     0.0     0.0     7.112 107.0
  6.0     0.0     1.0     0.0     7.762 115.0
  6.0     0.0     0.0     1.0     7.645 123.0
  6.0    -1.0    -1.0    -1.0     8.639 122.0
  7.0     1.0     0.0     0.0     7.667 128.0
  7.0     0.0     1.0     0.0     8.080 136.0
  7.0     0.0     0.0     1.0     6.678 140.0
  7.0    -1.0    -1.0    -1.0     6.739 122.0
  8.0     1.0     0.0     0.0     5.569 102.0
  8.0     0.0     1.0     0.0     5.049 103.0
  8.0     0.0     0.0     1.0     5.642  89.0
  8.0    -1.0    -1.0    -1.0     6.808  77.0
  9.0     1.0     0.0     0.0     6.636  89.0
  9.0     0.0     1.0     0.0     8.241  94.0
  9.0     0.0     0.0     1.0     7.968 104.0
  9.0    -1.0    -1.0    -1.0     8.044 108.0
 10.0     1.0     0.0     0.0     7.791 119.0
 10.0     0.0     1.0     0.0     7.024 126.0
 10.0     0.0     0.0     1.0     6.102 119.0
 10.0    -1.0    -1.0    -1.0     6.053 103.0
 11.0     1.0     0.0     0.0     5.941   0.0
 11.0     0.0     1.0     0.0     5.386   0.0
 11.0     0.0     0.0     1.0     5.811   0.0
 11.0    -1.0    -1.0    -1.0     6.716   0.0
 12.0     1.0     0.0     0.0     6.923   0.0
 12.0     0.0     1.0     0.0     6.939   0.0
 12.0     0.0     0.0     1.0     6.705   0.0
 12.0    -1.0    -1.0    -1.0     6.914   0.0
  0.0     0.0     0.0     0.0     0.1720  0.0
    0    0    0    0    2
    0    0    0    0    0
    0    0    0    0    2
    0    0    0    0    0
```

```
    0    0    0    0    1
    0    0    0    0    1
    0    0    0    0    4
  0.0     0.0     0.0     0.0     1.6743
  0.0     0.0     0.0     0.0    -0.9505
  0.0     0.0     0.0     0.0     1.4605
  0.0     0.0     0.0     0.0    -0.4862
  0.0     0.0     0.0     0.0     0.8993
```

10.3 Program Results

```
nag_tsa_multi_inp_model_forecast (g13bjc) Example Program Results

Parameters to g13bjc
____________________

nseries.....................   6

cfixed.................. Nag_TRUE
outfile.................    stdout

40 sets of observations were processed.

The residual mean square for the output series is    20.0902

The forecast values and their standard errors are


   i      fva      fsd

   1    93.398    4.4822
   2    96.958    6.1498
   3    86.046    7.0315
   4    77.589    7.2885
   5    82.139    7.3327
   6    96.276    7.5220
   7    98.345    8.0883
   8    93.577    8.8020

The values of z(t) and noise(t) are

   i       z1         z2         z3         z4         z5       noise

   1    -0.339     -3.889      0.000      0.000    188.603    -79.375
   2    -0.339     -0.000      4.514      0.000    199.438    -84.613
   3    -0.339     -0.000      0.000      2.479    204.683    -87.823
   4    -0.339      3.889     -4.514     -2.479    204.383    -91.940
   5    -0.678     -3.889      0.000      0.000    210.623    -89.056
   6    -0.678     -0.000      4.514      0.000    208.591    -77.426
   7    -0.678     -0.000      0.000      2.479    205.070    -80.870
   8    -0.678      3.889     -4.514     -2.479    203.407    -87.624
   9    -1.017     -3.889      0.000      0.000    206.974    -86.068
  10    -1.017     -0.000      4.514      0.000    206.132    -87.628
  11    -1.017     -0.000      0.000      2.479    201.920    -88.381
  12    -1.017      3.889     -4.514     -2.479    194.819    -75.698
  13    -1.356     -3.889      0.000      0.000    203.974    -76.729
  14    -1.356     -0.000      4.514      0.000    209.884    -75.041
  15    -1.356     -0.000      0.000      2.479    210.705    -76.828
  16    -1.356      3.889     -4.514     -2.479    210.373    -80.912
  17    -1.695     -3.889      0.000      0.000    205.942    -85.358
  18    -1.695     -0.000      4.514      0.000    194.575    -89.394
  19    -1.695     -0.000      0.000      2.479    185.866    -86.650
  20    -1.695      3.889     -4.514     -2.479    185.509    -84.709
  21    -2.035     -3.889      0.000      0.000    191.606    -78.682
  22    -2.035     -0.000      4.514      0.000    193.194    -80.673
  23    -2.035     -0.000      0.000      2.479    199.896    -77.340
  24    -2.035      3.889     -4.514     -2.479    203.497    -76.358
  25    -2.374     -3.889      0.000      0.000    214.552    -80.290
  26    -2.374     -0.000      4.514      0.000    213.770    -79.910
  27    -2.374     -0.000      0.000      2.479    216.796    -76.901
```

```
 28    -2.374      3.889     -4.514     -2.479    206.780    -79.302
 29    -2.713     -3.889      0.000      0.000    200.416    -91.814
 30    -2.713     -0.000      4.514      0.000    185.941    -84.742
 31    -2.713     -0.000      0.000      2.479    171.495    -82.261
 32    -2.713      3.889     -4.514     -2.479    166.673    -83.857
 33    -3.052     -3.889      0.000      0.000    173.418    -77.477
 34    -3.052     -0.000      4.514      0.000    176.573    -84.035
 35    -3.052     -0.000      0.000      2.479    192.594    -88.021
 36    -3.052      3.889     -4.514     -2.479    201.261    -87.105
 37    -3.391     -3.889      0.000      0.000    207.879    -81.599
 38    -3.391     -0.000      4.514      0.000    210.249    -85.372
 39    -3.391     -0.000      0.000      2.479    205.262    -85.350
 40    -3.391      3.889     -4.514     -2.479    193.874    -84.379
 41    -3.730     -3.889      0.000      0.000    185.617    -84.600
 42    -3.730      0.000      4.514      0.000    178.969    -82.795
 43    -3.730      0.000      0.000      2.479    169.607    -82.309
 44    -3.730      3.889     -4.514     -2.479    166.832    -82.409
 45    -4.069     -3.889      0.000      0.000    172.733    -82.636
 46    -4.069      0.000      4.514      0.000    178.579    -82.748
 47    -4.069      0.000      0.000      2.479    182.739    -82.804
 48    -4.069      3.889     -4.514     -2.479    183.582    -82.831
```

11 Optional Arguments

A number of optional input and output arguments to nag_tsa_multi_inp_model_forecast (g13bjc) are available through the structure argument **options** of type Nag_G13_Opt. An argument may be selected by assigning an appropriate value to the relevant structure member and those arguments not selected will be assigned default values. If no use is to be made of any of the optional arguments you should use the null pointer, `G13_DEFAULT`, in place of **options** when calling nag_tsa_multi_inp_model_forecast (g13bjc); 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 by direct assignment are checked within nag_tsa_multi_inp_model_forecast (g13bjc) for being within the required range, if outside the range, an error message is generated.

When all calls to nag_tsa_multi_inp_model_forecast (g13bjc) 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 Arguments 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_forecast (g13bjc) together with their default values where relevant.

`Boolean list` Nag_TRUE

`Boolean cfixed` Nag_FALSE

`double *zt`

`double *noise`

11.2 Description of the Optional Arguments

List – Nag_Boolean Default = Nag_TRUE

On entry: if **options**.**List** = Nag_TRUE then the argument settings which are used in the call to nag_tsa_multi_inp_model_forecast (g13bjc) will be printed.

cfixed – Nag_Boolean Default = Nag_FALSE

On entry: **options**.**cfixed** must be set to Nag_FALSE if the constant was estimated when the model was fitted, and Nag_TRUE if it was held at a fixed value. This only affects the degrees of freedom used in calculating the estimated residual variance.

zt – double * Default memory = (**nev** + **nfv**) × (**nseries** − 1)

On exit: this pointer is allocated memory internally with (**nev** + **nfv**) × (**nseries** − 1) elements corresponding to (**nev** + **nfv**) rows by **nseries** − 1 columns. The columns of **options**.**zt** hold the values of the input component series z_t.

noise – double * Default memory = **nev** + **nfv**

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