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_forcast(Nag_ArimaOrder *arimav, Integer nseries,
            Nag_TransfOrder *transfv, double para[], Integer npara,
            Integer nev, Integer nfv, double xxy[], Integer tdxxy,
            double rmsxy[], Integer mrx[], Integer tdmrx, double parx[],
            Integer ldparx, Integer tdparx, double fva[], double fsd[],
            Nag_G13_Opt *options, NagError *fail)
```

3. Description

The function 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 parameters remain unchanged in the supplied input series transfer function models and output noise series ARIMA model. The internal nuisance parameters 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 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) $z_t = \delta_1 z_{t-1} + \delta_2 z_{t-2} + \ldots + \delta_p z_{t-p} + \omega_0 x_{t-b} - \omega_1 x_{t-b-1} - \ldots - \omega_q x_{t-b-q}$.

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

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

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

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

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

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

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

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

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

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 routine 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 parameters and the estimated residual variance should be set to zero.

4. Parameters

arimav

Pointer to structure of type **Nag_ArimaOrder** with the following members:

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

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) parameters.

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

nseries

Input: the number of input and output series. There may be any number of input series (including none), but only one output series.
Constraints: **nseries** > 1 if there are no parameters in the model (that is $p = q = P = Q = 0$ and **options.cfixed** $=$ **TRUE**), **nseries** ≥ 1 otherwise.

transfv

Input: Pointer to structure of type Nag_TransfOrder with the following members:

b – Integer *
q – Integer *
p – Integer *
r – Integer *

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 parameters for input series i

are held in the ith element of the allocated memory for each pointer. **b**$[i-1]$ holds the value b_i, **q**$[i-1]$ holds the value q_i and **p**$[i-1]$ holds the value p_i.
For a simple input, $b_i = q_i = p_i = 0$.
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 parameters, and $r_i = 2$ suppresses this estimation. This choice may affect the returned forecasts.
When $r_i = 1$, any non-zero contents of the ith element of the memory of **b**, **q** and **p** are ignored.
Constraint: **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).

para[npara]

Input: estimates of the multi-input model parameters. These are in order firstly the ARIMA model parameters: p values of ϕ parameters, q values of θ parameters, P values of Φ parameters, Q values of Θ parameters.
These are followed by the transfer function model parameter 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.

npara

Input: the exact number of ϕ, θ, Φ, Θ, ω, δ, c parameters, 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.)

nev

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

nfv

Input: the number of forecast values of the output series required.
Constraint: **nfv** > 0.

xxy[nev+nfv][tdxxy]

Input: 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.

tdxxy

Input: the last dimension of array **xxy** as declared in the function from which nag_tsa_multi_inp_model_forecast is called.
Constraint: **tdxxy** $\geq$ **nseries**.

rmsxy[nseries]

Input: 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.
Output: **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.

mrx[7][tdmrx]

Input: 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 the null pointer (Integer $*$)0 may be supplied in place of **mrx**.

tdmrx

Input: the last dimension of array **mrx** as declared in the function from which nag_tsa_multi_inp_model_forecast is called.
Constraint: **tdmrx** $\geq$ **nseries** $- 1$.

parx[ldparx][tdparx]

Input: values of the parameters (ϕ, θ, Φ, and Θ) for each of the input series ARIMA models. Thus column i contains **mrx**[0][i] values of ϕ, **mrx**[2][i] values of θ, **mrx**[3][i] values of Φ and **mrx**[5][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 the null pointer (double *)0 may be supplied in place of **parx**.

ldparx

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

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

tdparx

Input: the last dimension of array **parx** as declared in the function from which nag_tsa_multi_inp_model_forecast is called.

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

fva[nfv]

Output: the required forecast values for the output series.

fsd[nfv]

Output: the standard errors for each of the forecast values.

options

Input/Output: a pointer to a structure of type Nag_G13_Opt whose members are optional parameters for nag_tsa_multi_inp_model_forecast. 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_forecast. Details of the optional parameters and their types are given below in Section 7.

fail

The NAG error parameter, see the Essential Introduction to the NAG C Library.

5. Error Indications and Warnings

A list of possible error exits from nag_tsa_multi_inp_model_forecast is given in Section 8.

6. Example

This example illustrates the use of the default option **G13_DEFAULT** in a call to nag_tsa_multi_inp_model_forecast. An example showing the use of optional parameters is given in Section 11. There is one example program file, the main program of which calls both examples. The main program is given below.

6.1 Example 1

This example illustrates the use of the default option **G13_DEFAULT** in a call to nag_tsa_multi_inp_model_forecast.

The data in the example relate to 40 observations of an output time series and 5 input time series. This example differs from Example 1 in nag_tsa_multi_inp_model_estim (g13bec) in that there are now 4 simple input series. The output series has one autoregressive (ϕ) parameter and one seasonal moving average (Θ) parameter. 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:

$$\phi = 0.495,\ \Theta = 0.238,\ \omega_1 = -0.367 \quad \omega_2 = -3.876 \quad \omega_3 = 4.516$$
$$\omega_4 = 2.474 \quad \omega_{5,1} = 8.629 \quad \delta_{5,1} = 0.688,\ c = -82.858.$$

A further 8 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 8 forecast values and their standard errors.

6.1.1. Program Text

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

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

#ifdef NAG_PROTO
static void ex1(void);
static void ex2(void);
#else
static void ex1();
static void ex2();
#endif

#define  NSERMX  6
#define  NPMAX  10
#define  LDPARX  8
#define  NFVMAX 10
#define  NEVMAX 40
#define  LDXXY   NEVMAX + NFVMAX

#define  TDMRX    NSERMX
#define  TDPARX   NSERMX
#define  TDXXY    NSERMX

main()
{
  /* Two examples are called, ex1() which uses the
   * default settings to solve the problem and
   * ex2() which solves the same problem with
   * some optional parameters set by the user.
   */

  Vprintf("g13bjc Example Program Results\n");
  Vscanf(" %*[^\n]"); /* Skip heading in data file */
  ex1();
  ex2();
  exit(EXIT_SUCCESS);
}

static void ex1()
{
  Integer  i, j, n, nev, nfv, npara, nseries, inser;
  double  fsd[NFVMAX], fva[NFVMAX], para[NFVMAX], parx[LDPARX][NSERMX],
  rmsxy[NSERMX], xxy[LDXXY][NSERMX];
  Integer mrx[7][NSERMX];
  Nag_ArimaOrder  arimav;
  Nag_TransfOrder transfv;
  static NagError fail;

  Vprintf("\ng13bjc example 1: no option setting.\n\n");

  /*  Skip heading in data file */
  Vscanf(" %*[^\n]");

  Vscanf("%ld%ld%ld", &nev, &nfv, &nseries);
```

```
if (nseries>0 && nseries<=NSERMX && nev>0 && nev<=NEVMAX &&
    nfv>0 && nfv<=NFVMAX)
  {
    /*
     * Allocate memory to the arrays in structure transfv containing
     * the transfer function model orders of the input series.
     */
    g13byc(nseries, &transfv, NAGERR_DEFAULT);

    /*
     * Read the orders vector of the ARIMA model for the output noise
     * component into structure arimav.
     */
    Vscanf("%ld%ld%ld%ld%ld%ld%ld", &arimav.p, &arimav.d, &arimav.q,
           &arimav.bigp, &arimav.bigd, &arimav.bigq, &arimav.s);
    /*
     * Read the transfer function model orders of the input series into
     * structure transfv.
     */
    inser = nseries - 1;

    for (j=0; j<inser; ++j)
      Vscanf("%ld", &transfv.b[j]);
    for (j=0; j<inser; ++j)
      Vscanf("%ld", &transfv.q[j]);
    for (j=0; j<inser; ++j)
      Vscanf("%ld", &transfv.p[j]);
    for (j=0; j<inser; ++j)
      Vscanf("%ld", &transfv.r[j]);

    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<=NPMAX)
      {
        for (i=0; i<npara; ++i)
          Vscanf("%lf", &para[i]);
        n = nev + nfv;
        for (i=0; i<n; ++i)
          for (j=0; j<nseries; ++j)
            Vscanf("%lf", &xxy[i][j]);
        for (i=0; i<nseries; ++i)
          Vscanf("%lf", &rmsxy[i]);
        for (i=0; i<7; ++i)
          for (j=0; j<inser; ++j)
            Vscanf("%ld", &mrx[i][j]);
        for (i=0; i<5; ++i)
          for (j=0; j<inser; ++j)
            Vscanf("%lf", &parx[i][j]);

        fail.print = TRUE;

        g13bjc(&arimav, nseries, &transfv, para, npara, nev, nfv,
               (double *)xxy, (Integer)TDXXY, rmsxy, (Integer *)mrx,
               (Integer)TDMRX, (double *)parx, (Integer)LDPARX,
               (Integer)TDPARX, fva, fsd, G13_DEFAULT, &fail);

        if (fail.code==NE_NOERROR || fail.code==NE_SOLUTION_FAIL_CONV ||
            fail.code==NE_MAT_NOT_POS_DEF)
          {
            Vprintf("\nThe residual mean square for the output\n");
            Vprintf("series is also derived and its value is %10.4f\n\n",
                    rmsxy[nseries-1]);
            Vprintf("The forecast values and their standard errors are\n\n");
            Vprintf("\n   i      fva      fsd\n\n");
            for (i=0; i<nfv; ++i)
              Vprintf("%4ld%10.3f%10.4f\n", i+1, fva[i], fsd[i]);
          }
```

```
        }
      else
        {
          Vfprintf(stderr, "npara is out of range: npara = %-3ld\n", npara);
          g13bzc(&transfv);
          exit(EXIT_FAILURE);
        }
    }
  else
    {
      Vfprintf(stderr, "One or more of nseries, nev and nfv are out of \
range: nseries = %-3ld,  nev = %-3ld while  nfv = %-3ld\n", nseries, nev, nfv);
      exit(EXIT_FAILURE);
    }
  g13bzc(&transfv);
  if (fail.code!=NE_NOERROR)
    exit(EXIT_FAILURE);
}
```

6.1.2. Program Data

```
g13bjc Example Program Data

Example 1 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
```

6.1.3. Program Results

```
g13bjc Example Program Results

g13bjc example 1: no option setting.


Parameters to g13bjc
--------------------

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

cfixed..................... FALSE

The residual mean square for the output
series is also derived and its value is    20.7599

The forecast values and their standard errors are


   i      fva      fsd

   1    93.398    4.5563
   2    96.958    6.2172
   3    86.046    7.0933
   4    77.589    7.3489
   5    82.139    7.3941
   6    96.276    7.5823
   7    98.345    8.1445
   8    93.577    8.8536
```

7. Optional Parameters

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

Before assigning values to **options** the structure must be initialised 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 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 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**.

7.1. 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_forecast together with their default values where relevant.

`Boolean list`	**TRUE**
`Boolean cfixed`	**FALSE**
`double *zt`	
`double *noise`	

7.2. Description of Optional Parameters

list – Boolean Default = **TRUE**

Input: If **options.list** = **TRUE** then the parameter settings which are used in the call to nag_tsa_multi_inp_model_forecast will be printed.

cfixed – Boolean Default = **FALSE**

Input: **cfixed** must be set to **FALSE** if the constant was estimated when the model was fitted, and **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)

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

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

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

8. Error Indications

NE_G13_OPTIONS_NOT_INIT

On entry, the option structure, **options**, has not been initialised 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 initialised.

NE_INT_ARRAY_2

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

NE_BAD_PARAM

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

NE_INT_ARG_LT

On entry, **nseries** must not be less than 1: **nseries** = ⟨*value*⟩.

NE_INT_ARG_LE

On entry, **nfv** must not be less than or equal to 0: **nfv** = ⟨*value*⟩.

NE_2_INT_ARG_LT

On entry, **tdxxy** = ⟨*value*⟩ while **nseries** = ⟨*value*⟩. These parameters must satisfy **tdxxy** ≥ **nseries**.
On entry, **tdmrx** = ⟨*value*⟩ while **nseries** − 1 = ⟨*value*⟩. These parameters must satisfy **tdmrx** ≥ **nseries** − 1.
On entry, **ldparx** = ⟨*value*⟩ while *nce* = ⟨*value*⟩. These parameters must satisfy **ldparx** ≥ *nce*. (See the expression for *nce* in Section 4 where **ldparx** is described).
On entry, **tdparx** = ⟨*value*⟩ while **nseries** − 1 = ⟨*value*⟩. These parameters must satisfy **tdparx** ≥ **nseries** − 1.

NE_ALLOC_FAIL
Memory allocation failed.

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

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

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

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

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.

9. Further Comments

The time taken by the function is approximately proportional to the product of the length of each series and the square of the number of parameters in the multi-input model.

9.1 Accuracy

The computation used is believed to be stable.

9.2 References

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

10. See Also

nag_tsa_multi_inp_model_estim (g13bec)
nag_tsa_options_init (g13bxc)
nag_tsa_transf_orders (g13byc)
nag_tsa_trans_free (g13bzc)
nag_tsa_free (g13xzc)

11. Example 2

This example illustrates the use of the **options** parameter in a call to nag_tsa_multi_inp_model_forecast.

The data in the example relate to the same 40 obervations of an output time series and 5 input time series as in Example 1. This example differs from Example 2 in nag_tsa_multi_inp_model_estim (g13bec) in that there are now 4 simple input series. The output series has one autoregressive (ϕ) parameter and one seasonal moving average (Θ) parameter. 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:

$$\phi = 0.495,\ \Theta = 0.238,\ \omega_1 = -0.367 \quad \omega_2 = -3.876 \quad \omega_3 = 4.516$$
$$\omega_4 = 2.474 \quad \omega_{5,1} = 8.629 \quad \delta_{5,1} = 0.688,\ c = -82.858.$$

A further 8 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 8 forecast values and their standard errors, and the values of the components z_t and the output noise component n_t.

11.1. Program Text

```
static void ex2()
{
  Integer  i, j, n, nev, nfv, npara, nseries, inser;
  double  fsd[NFVMAX], fva[NFVMAX], para[NFVMAX], parx[LDPARX][TDPARX],
  rmsxy[NSERMX], xxy[LDXXY][TDXXY];
  Integer mrx[7][NSERMX];
  Nag_ArimaOrder  arimav;
  Nag_TransfOrder transfv;
  Nag_G13_Opt  options;
  static NagError fail;

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

  Vprintf("\n\ng13bjc example 2: using option setting.\n\n");

  /*  Skip heading in data file */
  Vscanf(" %*[^\n]");

  /*
   * Initialise the option-setting function.
   */
  g13bxc(&options);

  Vscanf("%ld%ld%ld", &nev, &nfv, &nseries);
  if (nseries>0 && nseries<=NSERMX && nev>0 && nev<=NEVMAX &&
      nfv>0 && nfv<=NFVMAX)
    {
      /*
       * Set option variable to the desired value.
       */

      options.cfixed = TRUE;
      /*
       * Allocate memory to the arrays in structure transfv containing
       * the transfer function model orders of the input series.
       */
      g13byc(nseries, &transfv, NAGERR_DEFAULT);

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

```
      for (j=0; j<inser; ++j)
        Vscanf("%ld", &transfv.b[j]);
      for (j=0; j<inser; ++j)
        Vscanf("%ld", &transfv.q[j]);
      for (j=0; j<inser; ++j)
        Vscanf("%ld", &transfv.p[j]);
      for (j=0; j<inser; ++j)
        Vscanf("%ld", &transfv.r[j]);

      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<=NPMAX)
        {
          for (i=0; i<npara; ++i)
            Vscanf("%lf", &para[i]);
          n = nev + nfv;
          for (i=0; i<n; ++i)
            for (j=0; j<nseries; ++j)
              Vscanf("%lf", &xxy[i][j]);
          for (i=0; i<nseries; ++i)
            Vscanf("%lf", &rmsxy[i]);
          for (i=0; i<7; ++i)
            for (j=0; j<inser; ++j)
              Vscanf("%ld", &mrx[i][j]);
          for (i=0; i<5; ++i)
            for (j=0; j<inser; ++j)
              Vscanf("%lf", &parx[i][j]);

          fail.print = TRUE;

          g13bjc(&arimav, nseries, &transfv, para, npara, nev, nfv,
                 (double *)xxy, (Integer)TDXXY, rmsxy, (Integer *)mrx,
                 (Integer)TDMRX, (double *)parx, (Integer)LDPARX,
                 (Integer)TDPARX, fva, fsd, &options, &fail);

          if (fail.code==NE_NOERROR || fail.code==NE_SOLUTION_FAIL_CONV ||
              fail.code==NE_MAT_NOT_POS_DEF)
            {
              Vprintf("%1ld sets of observations were processed.\n",nev);
              Vprintf("\nThe residual mean square for the output ");
              Vprintf("series is %10.4f\n\n", rmsxy[nseries-1]);
              Vprintf("The forecast values and their standard errors are\n\n");
              Vprintf("\n   i      fva      fsd\n\n");
              for (i=0; i<nfv; ++i)
                Vprintf("%4ld%10.3f%10.4f\n", i+1, fva[i], fsd[i]);
              Vprintf("\nThe values of z(t) and noise(t) are\n\n");
              Vprintf("   i       z1         z2         z3         z4\
   z5       noise\n\n");
              for (i=0; i<n; ++i)
                {
                  Vprintf("%4ld", i+1);
                  for (j=0; j<nseries-1; ++j)
                    Vprintf("%10.3f ", ZT(i,j));
                  Vprintf("%10.3f\n", options.noise[i]);
                }
            }
        }
      else
        {
          Vfprintf(stderr, "npara is out of range: npara = %-3ld\n", npara);
          g13xzc(&options);
          g13bzc(&transfv);
          exit(EXIT_FAILURE);
        }
    }
  else
```

```
    {
      Vfprintf(stderr, "One or more of nseries, nev and nfv are out of range:\
 nseries = %-3ld,  nev = %-3ld while  nfv = %-3ld\n", nseries, nev, nfv);
      exit(EXIT_FAILURE);
    }
  g13xzc(&options);
  g13bzc(&transfv);
  if (fail.code!=NE_NOERROR)
    exit(EXIT_FAILURE);
}
```

11.2. Program Data

```
Example 2 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
```

11.3. Program Results

```
g13bjc example 2: using option setting.


Parameters to g13bjc
--------------------

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

cfixed..................... TRUE
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