

NAG Library Function Document

nag_tsa_multi_inp_update (g13bgc)

1 Purpose

nag_tsa_multi_inp_update (g13bgc) accepts a series of new observations of an output time series and any associated input time series, for which a multi-input model is already fully specified, and updates the ‘state set’ information for use in constructing further forecasts.

The previous specification of the multi-input model will normally have been obtained by using nag_tsa_multi_inp_model_estim (g13bec) to estimate the relevant transfer function and ARIMA parameters. The supplied state set will originally have been produced by nag_tsa_multi_inp_model_estim (g13bec) (or possibly nag_tsa_multi_inp_model_forecast (g13bjc)), but may since have been updated by nag_tsa_multi_inp_update (g13bgc).

2 Specification

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

void nag_tsa_multi_inp_update (Nag_ArimaOrder *arimav, Integer nser,
    Nag_TransfOrder *transfv, const double para[], Integer npara,
    Integer nnv, double xxyn[], Integer tdxyn, Integer kzef,
    Nag_G13_Opt *options, Nag_Error *fail)
```

3 Description

The multi-input model is specified in Section 3 in nag_tsa_multi_inp_model_estim (g13bec). The form of these equations required to update the state set is as follows:

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

the transfer models which generate input component values $z_{i,t}$ from one or more inputs $x_{i,t}$,

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

which generates the output noise component from the output y_t and the input components, and

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

the ARIMA model for the output noise which generates the residuals a_t .

The state set (as also given in Section 3 in nag_tsa_multi_inp_model_estim (g13bec)) is the collection of terms

$$z_{n+1-k}, x_{n+1-k}, n_{n+1-k}, w_{n+1-k}, e_{n+1-k} \quad \text{and} \quad a_{n+1-k}$$

for $k = 1$ up to the maximum lag associated with each of these series respectively, in the above model equations. n is the latest time point of the series from which the state set has been generated.

The function accepts further values of the series $y_t, x_{1,t}, x_{2,t}, \dots, x_{m,t}$, for $t = n + 1, \dots, n + l$, and applies the above model equations over this time range, to generate new values of the various model components, noise series and residuals. The state set is reconstructed, corresponding to the latest time point $n + l$, the earlier values being discarded.

The set of residuals corresponding to the new observations may be of use in checking that the new observations conform to the previously fitted model. The components of the new observations of the output series which are due to the various inputs, and the noise component, are also optionally returned.

The parameters of the model are not changed in this function.

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	<i>Input</i>
q – Integer	<i>Input</i>
bigp – Integer	<i>Input</i>
bigd – Integer	<i>Input</i>
bigq – Integer	<i>Input</i>
s – Integer	<i>Input</i>

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) parameters.

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

2: **nser** – Integer *Input*

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

3: **transfv** – Nag_TransfOrder *

Pointer to structure of type Nag_TransfOrder with the following members:

b – Integer *	<i>Input</i>
q – Integer *	<i>Input</i>
p – Integer *	
r – Integer *	<i>Input</i>

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 i th element of the allocated memory for each pointer. **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, $b_i = q_i = p_i = 0$.

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 nonzero contents of the i th element of the memory of **b**, **transfv**→**q** and **transfv**→**p** are ignored.

Constraint: **r**[i – 1] = 1, 2 or 3, for $i = 1, 2, \dots, \mathbf{nseries} - 1$

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

- 4: **para**[**npara**] – const double *Input*
On entry: estimates of the multi-input model parameters as returned by `nag_tsa_multi_inp_model_estim` (g13bec). These are in order, firstly the ARIMA model parameters: p values of ϕ parameters, q values of θ parameters, P values of Φ parameters and Q values of Θ parameters. These are followed by the transfer function model parameter values $\omega_0, \omega_1, \dots, \omega_{q_1}, \delta_1, \delta_2, \dots, \delta_{p_1}$ for the first of any input series and similarly for each subsequent input series. The final component of **para** is the value of the constant c .
- 5: **npara** – Integer *Input*
On entry: the exact number of $\phi, \theta, \Phi, \Theta, \omega, \delta$ and c parameters. (c must be included whether its value was previously estimated or was set fixed.)
- 6: **nnv** – Integer *Input*
On entry: the number of new observation sets being used to update the state set, each observation set consisting of a value of the output series and the associated values of each of the input series at a particular time point.
- 7: **xxyn**[**nnv** × **tdxxyn**] – double *Input/Output*
Note: the (i, j) th element of the matrix is stored in **xxyn**[($i - 1$) × **tdxxyn** + $j - 1$].
On entry: the **nnv** new observation sets being used to update the state set. Column $i - 1$ contains the values of input series i , for $i = 1, 2, \dots, \mathbf{nser} - 1$. Column **nser** - 1 contains the values of the output series. Consecutive rows correspond to increasing time sequence.
On exit: if **kzef** = 0, **xxyn** remains unchanged.
 If **kzef** ≠ 0, the columns of **xxyn** hold the corresponding values of the input component series z_t and the output noise component n_t in that order.
- 8: **tdxxyn** – Integer *Input*
On entry: the stride separating matrix column elements in the array **xxyn**.
Constraint: **tdxxyn** ≥ **nser**.
- 9: **kzef** – Integer *Input*
On entry: must not be set to 0, if the values of the input component series z_t and the values of the output noise component n_t are to overwrite the contents of **xxyn** on exit, and must be set to 0 if **xxyn** is to remain unchanged on exit.
- 10: **options** – Nag_G13_Opt * *Input/Output*
On entry: a pointer to a structure of type Nag_G13_Opt as returned by `nag_tsa_multi_inp_model_estim` (g13bec) or `nag_tsa_multi_inp_model_forecast` (g13bjc).
On exit: the structure contains the updated state space information.
- 11: **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

NE_ALLOC_FAIL

Dynamic memory allocation failed.

See Section 2.3.1.2 in How to Use the NAG Library and its Documentation for further information.

NE_BAD_PARAM

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

NE_INT

On entry, $\mathbf{tdxxyn} = \langle value \rangle$.

Constraint: $\mathbf{tdxxyn} > 0$.

NE_INT_2

On entry, $\mathbf{tdxxyn} = \langle value \rangle$ and $\mathbf{nser} = \langle value \rangle$.

Constraint: $\mathbf{tdxxyn} \geq \mathbf{nser}$.

NE_INTERNAL_ERROR

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

An unexpected error has been triggered by this function. Please contact NAG.

See Section 2.7.6 in How to Use the NAG Library and its Documentation for further information.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.

See Section 2.7.5 in How to Use the NAG Library and its Documentation for further information.

NE_STRUCT_CORRUPT

Values of the members of structures **arimav**, **transfv** and **options** are not compatible.

7 Accuracy

The computations are believed to be stable.

8 Parallelism and Performance

`nag_tsa_multi_inp_update` (g13bgc) is not threaded in any implementation.

9 Further Comments

The time taken by `nag_tsa_multi_inp_update` (g13bgc) is approximately proportional to $\mathbf{nnv} \times \mathbf{npara}$.

10 Example

This example uses the data described in `nag_tsa_multi_inp_model_estim` (g13bec) in which 40 observations of an output time series and a single input series were processed. In this example a model which included seasonal differencing of order 1 was used. The 10 values of the state set and the 5 final values of **para** then obtained are used as input to this program, together with the values of 4 new observations and the transfer function orders of the input series. The model used is $\phi_1 = 0.5158$, $\theta_1 = 0.9994$, $\omega_0 = 8.6343$, $\delta_1 = 0.6726$, $c = -0.3172$.

The following are computed and printed out: the updated state set, the residuals a_t and the values of the components z_t and the output noise component n_t corresponding to the new observations.

10.1 Program Text

```

/* nag_tsa_multi_inp_update (g13bgc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

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

int main(void)
{
    double df, objf, rss;
    Integer exit_status = 0, i, inser, j, kzef, nnv, npara, nser,
           nxy, tdxy, tdxyyn;
    double *para = 0, *sd = 0, *xxy = 0, *xxyyn = 0;
    /* Nag types */
    Nag_ArimaOrder arimav;
    Nag_TransfOrder transfv;
    Nag_G13_Opt options;
    NagError fail;

#define XXY(I, J) xxy[(I - 1) * tdxy + J - 1]
#define XXYN(I, J) xxyyn[(I - 1) * tdxyyn + J - 1]

    INIT_FAIL(fail);

    printf("nag_tsa_multi_inp_update (g13bgc) Example Program Results\n");

    /* Skip heading in data file */
#ifdef _WIN32
    scanf_s("%*[\n]");
#else
    scanf("%*[\n]");
#endif
    /* 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 "%" NAG_IFMT "", &nxy,
           &nser, &options.max_iter, &nnv);
#else
    scanf("%" NAG_IFMT "%" NAG_IFMT "%" NAG_IFMT "%" NAG_IFMT "", &nxy, &nser,
           &options.max_iter, &nnv);
#endif

    if (nxy > 0 && nser > 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 (g13bxc).
         * Allocates memory to transfer function model orders
         */
        nag_tsa_transf_orders(nser, &transfv, &fail);
        if (fail.code != NE_NOERROR) {
            printf("Error from nag_tsa_transf_orders (g13bxc).\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 "" 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 "" 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 = nser - 1;
for (j = 1; j <= inser; ++j) {
#ifdef _WIN32
    scanf_s("%" NAG_IFMT "", &transfv.b[j - 1]);
#else
    scanf("%" NAG_IFMT "", &transfv.b[j - 1]);
#endif
}
for (j = 1; j <= inser; ++j) {
#ifdef _WIN32
    scanf_s("%" NAG_IFMT "", &transfv.q[j - 1]);
#else
    scanf("%" NAG_IFMT "", &transfv.q[j - 1]);
#endif
}
for (j = 1; j <= inser; ++j) {
#ifdef _WIN32
    scanf_s("%" NAG_IFMT "", &transfv.p[j - 1]);
#else
    scanf("%" NAG_IFMT "", &transfv.p[j - 1]);
#endif
}
for (j = 1; j <= inser; ++j) {
#ifdef _WIN32
    scanf_s("%" NAG_IFMT "", &transfv.r[j - 1]);
#else
    scanf("%" NAG_IFMT "", &transfv.r[j - 1]);
#endif
}
#ifdef _WIN32
    scanf_s("%*[\n]");
#else
    scanf("%*[\n]");
#endif

npara = 0;
for (i = 1; i <= inser; ++i) {
    npara += transfv.q[i - 1] + transfv.p[i - 1];
}
npara += arimav.p + arimav.q + arimav.bigp + arimav.bigq + nser;

tdxxy = nser;
tdxxyn = nser;
/* Memory allocation */
if (!(para = NAG_ALLOC(npara, double)) ||
    !(sd = NAG_ALLOC(npara, double)) ||
    !(xxy = NAG_ALLOC(nxxy * tdxxy, double)) ||
    !(xxyn = NAG_ALLOC(nnv * tdxxyn, double)))
{
    printf("Memory allocation failure\n");
    exit_status = -1;
}

```

```

        goto END;
    }

    for (i = 1; i <= npara; ++i) {
#ifdef _WIN32
        scanf_s("%lf", &para[i - 1]);
#else
        scanf("%lf", &para[i - 1]);
#endif
    }
#ifdef _WIN32
    scanf_s("%*[\n]");
#else
    scanf("%*[\n]");
#endif

    for (i = 1; i <= nxy; ++i) {
        for (j = 1; j <= nser; ++j) {
#ifdef _WIN32
            scanf_s("%lf", &XXY(i, j));
#else
            scanf("%lf", &XXY(i, j));
#endif
        }
    }
#ifdef _WIN32
    scanf_s("%*[\n]");
#else
    scanf("%*[\n]");
#endif

    for (i = 1; i <= nnv; ++i) {
        for (j = 1; j <= nser; ++j) {
#ifdef _WIN32
            scanf_s("%lf", &XXYN(i, j));
#else
            scanf("%lf", &XXYN(i, j));
#endif
        }
    }
#ifdef _WIN32
    scanf_s("%*[\n]");
#else
    scanf("%*[\n]");
#endif

    options.print_level = Nag_NoPrint;
    /* nag_tsa_multi_inp_model_estim (g13bec).
     * Estimation for time series models
     */
    fflush(stdout);
    nag_tsa_multi_inp_model_estim(&arimav, nser, &transfv, para, npara, nxy,
                                  xxy, tdxxy, sd, &rss, &objf, &df, &options,
                                  &fail);

    if (fail.code != NE_NOERROR) {
        printf("\nError from nag_tsa_multi_inp_model_estim "
              "(g13bec).\n%s\n.", fail.message);
        exit_status = 1;
        goto END;
    }

    /* Calculate update */
    kzef = 1;
    /* nag_tsa_multi_inp_update (g13bgc).
     * Multivariate time series, update state set for
     * forecasting from multi-input model
     */
    nag_tsa_multi_inp_update(&arimav, nser, &transfv, para, npara, nnv, xxyn,
                              tdxxy, kzef, &options, &fail);

```

```

if (fail.code != NE_NOERROR) {
    printf("\nError from "
           "nag_tsa_multi_inp_update (g13bgc).\n%s\n.", fail.message);
    exit_status = 1;
    goto END;
}

printf("\n\nThe residuals (after differencing)\n");
for (i = 1; i <= nnv; i++) {
    printf("%2" NAG_IFMT "%12.4f\n", i, options.res[i - 1]);
}
printf("\n\n");

printf("\nThe values of z(t) and n(t)\n");
for (i = 1; i <= nnv; i++) {
    printf("%2" NAG_IFMT "", i);
    for (j = 1; j <= nser; j++) {
        printf(" %9.4f", XXYN(i, j));
    }
    printf("\n");
}
printf("\n");
}
else {
    if (nxyy <= 0 || nser <= 0) {
        printf("One or both of nxyy and nser are out of range: "
               "nxyy = %-3" NAG_IFMT " while nser = %-3" NAG_IFMT "\n",
               nxyy, nser);
        exit_status = -1;
        goto END;
    }
}
}

END:
/* nag_tsa_trans_free (g13bzc).
 * Freeing function for the structure holding the transfer
 * function model orders
 */
nag_tsa_trans_free(&transfv);
/* nag_tsa_free (g13xzc).
 * Freeing function for use with g13 option setting
 */
nag_tsa_free(&options);
NAG_FREE(para);
NAG_FREE(sd);
NAG_FREE(xxy);
NAG_FREE(xxyn);

return exit_status;
}

```

10.2 Program Data

```

nag_tsa_multi_inp_update (g13bgc) Example Program Data
 40      2      20      4
 1      0      0      0      1      1      4
 1
 0
 1
 3
0.5158      0.9994      8.6343      0.6726      -0.3172
 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 : End of initial data
5.9410  96.0000
5.3860  95.0000
5.8110  80.0000
6.7160  88.0000 : End of update data

```

10.3 Program Results

nag_tsa_multi_inp_update (g13bgc) 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_NoPrint

```

The residuals (after differencing)

```

1      1.4649
2     -2.4577
3     -4.7624
4     13.2904

```

The values of $z(t)$ and $n(t)$

```

1  176.3560 -80.3560
2  169.9140 -74.9140
3  160.7891 -80.7891
4  158.3212 -70.3212

```