NAG Library Function Document

nag_tsa_transf_prelim_fit (g13bdc)

1 Purpose

nag_tsa_transf_prelim_fit (g13bdc) calculates preliminary estimates of the parameters of a transfer function model.

2 Specification

```
#include <nag.h>
#include <nagg13.h>
void nag_tsa_transf_prelim_fit (double r0, const double r[], Integer nl,
     Nag_TransfOrder *transfv, double s, double wds[], Integer isf[],
     NagError *fail)
```

3 Description

nag_tsa_transf_prelim_fit (g13bdc) calculates estimates of parameters $\delta_1, \delta_2, \ldots, \delta_p$, $\omega_0, \omega_1, \ldots, \omega_q$ in the transfer function model

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

given cross-correlations between the series x_t and lagged values of y_t:

$$r_{xy}(l), \quad l = 0, 1, \ldots, L$$

and the ratio of standard deviations s_y/s_x, as supplied by nag_tsa_cross_corr (g13bcc).

It is assumed that the series x_t used to calculate the cross-correlations is a sample from a time series with true autocorrelations of zero. Otherwise the cross-correlations between the series b_t and a_t, as defined in the description of nag_tsa_arma_filter (g13bac), should be used in place of those between y_t and x_t.

The estimates are obtained by solving for $\delta_1, \delta_2, \ldots, \delta_p$ the equations

$$r_{xy}(b+q+j) = \delta_1 r_{xy}(b+q+j-1) + \cdots + \delta_p r_{xy}(b+q+j-p), \quad j = 1, 2, \ldots, p$$

then calculating

$$\omega_i = \pm\big(s_y/s_x\big)\big[r_{xy}(b+i) - \delta_1 r_{xy}(b+i-1) - \cdots - \delta_p r_{xy}(b+i-p)\big], \quad i = 0, 1, \ldots, q$$

where the '$+$' is used for ω_0 and '$-$' for ω_i, $i > 0$.

Any value of $r_{xy}(l)$ arising in these equations for $l < b$ is taken as zero. The parameters $\delta_1, \delta_2, \ldots, \delta_p$ are checked as to whether they satisfy the stability criterion.

4 References

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

5 Arguments

1: **r0** – double *Input*

On entry: the cross-correlation between the two series at lag 0, $r_{xy}(0)$.

Constraint: $-1.0 \leq$ **r0** ≤ 1.0.

2: **r**[**nl**] – const double *Input*

On entry: the cross-correlations between the two series at lags 1 to L, $r_{xy}(l)$, for $l = 1, 2, \ldots, L$.

Constraint: $-1.0 \leq \mathbf{r}[i] \leq 1.0$, for $i = 0, 1, \ldots, \mathbf{nl} - 1$.

3: **nl** – Integer *Input*

On entry: L, the number of lagged cross-correlations in the array **r**.

Constraint: $\mathbf{nl} \geq \max(\mathbf{transfv.nag\_b} + \mathbf{transfv.nag\_q} + \mathbf{transfv.nag\_p}, 1)$.

4: **transfv** – Nag_TransfOrder * *Input*

On entry: the orders of the transfer function model where the triplet (**transfv.nag_b**, **transfv.nag_q**, **transfv.nag_p**) corresponds to the triplet (b, q, p) as described in Section 2.3.1 in the g13 Chapter Introduction.

Constraints:

$\mathbf{transfv.nag\_b} \geq 0$;
$\mathbf{transfv.nag\_q} \geq 0$;
$\mathbf{transfv.nag\_p} \geq 0$.

5: **s** – double *Input*

On entry: the ratio of the standard deviation of the y series to that of the x series, s_y/s_x.

Constraint: $\mathbf{s} > 0.0$.

6: **wds**[dim] – double *Output*

Note: the dimension, dim, of the array **wds** must be at least $(\mathbf{transfv.nag\_q} + \mathbf{transfv.nag\_p} + 1)$.

On exit: the preliminary estimates of the parameters of the transfer function model in the order of $q + 1$ MA-like parameters followed by the p AR-like parameters. If the estimation of either type of parameter fails then these arguments are set to 0.0.

7: **isf**[**2**] – Integer *Output*

On exit: indicators of the success of the estimation of MA-like and AR-like parameters respectively. A value 0 indicates that there are no parameters of that type to be estimated. A value of 1 or -1 indicates that there are parameters of that type in the model and the estimation of that type has been successful or unsuccessful respectively. Note that there is always at least one MA-like parameter in the model.

8: **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 ⟨*value*⟩ had an illegal value.

NE_CONSTRAINT

On entry, **transfv** = ⟨*value*⟩.
Constraint: **transfv**.**nag_b** ≥ 0.

On entry, **transfv** = ⟨*value*⟩.
Constraint: **transfv**.**nag_p** ≥ 0.

On entry, **transfv** = ⟨*value*⟩.
Constraint: **transfv**.**nag_q** ≥ 0.

NE_INT_4

On entry, **nl** = ⟨*value*⟩, **transfv**.**nag_b** = ⟨*value*⟩, **transfv**.**nag_q** = ⟨*value*⟩ and **transfv**.**nag_p** = ⟨*value*⟩.
Constraint: **nl** ≥ max(**transfv**.**nag_b** + **transfv**.**nag_q** + **transfv**.**nag_p**, 1).

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_REAL

On entry, **r0** lies outside [−1.0, 1.0]: **r0** = ⟨*value*⟩.

On entry, **s** = ⟨*value*⟩.
Constraint: **s** > 0.0.

NE_REAL_ARRAY_ELEM_CONS

On entry, **r**[*I* − 1] lies outside [−1.0, 1.0]: *I* = ⟨*value*⟩ and **r**[*I* − 1] = ⟨*value*⟩.

7 Accuracy

Equations used in the computations may become unstable, in which case results are reset to zero with array **isf** values set accordingly.

8 Parallelism and Performance

nag_tsa_transf_prelim_fit (g13bdc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_tsa_transf_prelim_fit (g13bdc) makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the x06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this function. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

The time taken by nag_tsa_transf_prelim_fit (g13bdc) is roughly proportional to (**transfv**.**nag_q** + **transfv**.**nag_p** + 1)^3.

10 Example

This example reads the cross-correlations between two series at lags 0 to 6. It then reads a $(3,2,1)$ transfer function model and calculates and prints the preliminary estimates of the parameters of the model.

10.1 Program Text

```
/* nag_tsa_transf_prelim_fit (g13bdc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

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

int main(void)
{
  /* Scalars */
  double r0, s;
  Integer exit_status, i, iwa, nl, nwds;

  /* Arrays */
  double *r = 0, *wa = 0, *wds = 0;
  Integer isf[3];
  Nag_TransfOrder transfv;
  NagError fail;

  INIT_FAIL(fail);

  exit_status = 0;

  printf("nag_tsa_transf_prelim_fit (g13bdc) Example Program Results\n");

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

#ifdef _WIN32
  scanf_s("%" NAG_IFMT "%*[^\n] ", &nl);
#else
  scanf("%" NAG_IFMT "%*[^\n] ", &nl);
#endif
#ifdef _WIN32
  scanf_s("%lf%*[^\n] ", &r0);
#else
  scanf("%lf%*[^\n] ", &r0);
#endif

  if (nl > 0) {
    /* Allocate array r */
    if (!(r = NAG_ALLOC(nl, double)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

    for (i = 1; i <= nl; ++i)
#ifdef _WIN32
```

```
      scanf_s("%lf", &r[i - 1]);
#else
      scanf("%lf", &r[i - 1]);
#endif
#ifdef _WIN32
    scanf_s("%*[^\n] ");
#else
    scanf("%*[^\n] ");
#endif

#ifdef _WIN32
    scanf_s("%ld %ld %ld", &transfv.nag_b, &transfv.nag_q, &transfv.nag_p);
#else
    scanf("%ld %ld %ld", &transfv.nag_b, &transfv.nag_q, &transfv.nag_p);
#endif
#ifdef _WIN32
    scanf_s("%lf%*[^\n] ", &s);
#else
    scanf("%lf%*[^\n] ", &s);
#endif

    nwds = transfv.nag_q + transfv.nag_p + 1;
    iwa = transfv.nag_p * (transfv.nag_p + 1);

    /* Allocate arrays wa and wds */
    if (!(wa = NAG_ALLOC(iwa, double)) || !(wds = NAG_ALLOC(nwds, double)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

    /* nag_tsa_transf_prelim_fit (g13bdc).
     * Multivariate time series, preliminary estimation of
     * transfer function model
     */
    nag_tsa_transf_prelim_fit(r0, r, nl, &transfv, s, wds, isf, &fail);
    if (fail.code != NE_NOERROR) {
      printf("Error from nag_tsa_transf_prelim_fit (g13bdc).\n%s\n",
             fail.message);
      exit_status = 1;
      goto END;
    }

    printf("\n");
    printf("Success/failure indicator%4" NAG_IFMT "%4" NAG_IFMT "\n",
           isf[0], isf[1]);
    printf("\n");
    printf("Transfer function model B, Q, P =");

    printf("%4ld %4ld %4ld\n", transfv.nag_b, transfv.nag_q, transfv.nag_p);

    printf("\n");
    printf("Parameter initial estimates\n");

    for (i = 1; i <= nwds; ++i)
      printf("%10.4f", wds[i - 1]);
    printf("\n");
  }

END:
  NAG_FREE(r);
  NAG_FREE(wa);
  NAG_FREE(wds);

  return exit_status;
}
```

10.2 Program Data

```
nag_tsa_transf_prelim_fit (g13bdc) Example Program Data
     6
 -0.0155
  0.0339 -0.0374 -0.2895 -0.3430 -0.4518 -0.2787
     3     2     1
  1.9256
```

10.3 Program Results

```
nag_tsa_transf_prelim_fit (g13bdc) Example Program Results

Success/failure indicator   1   1

Transfer function model B, Q, P =   3    2    1

Parameter initial estimates
   -0.5575    0.3166    0.4626    0.6169
```