

## NAG C Library Function Document

### nag\_tsa\_multi\_diff (g13dlc)

#### 1 Purpose

nag\_tsa\_multi\_diff (g13dlc) differences and/or transforms a multivariate time series.

#### 2 Specification

```
void nag_tsa_multi_diff (Integer k, Integer n, const double z[], const Integer tr[],
    const Integer id[], const double delta[], double w[], Integer *nd,
    NagError *fail)
```

#### 3 Description

For certain time series it may first be necessary to difference the original data to obtain a stationary series before calculating autocorrelations, etc. This routine also allows the user to apply either a square root or a log transformation to the original time series to stabilize the variance if required.

If the order of differencing required for the  $i$ th series is  $d_i$ , then the differencing operator is defined by  $\delta_i(B) = 1 - \delta_{i1}B - \delta_{i2}B^2 - \dots - \delta_{id_i}B^{d_i}$ , where  $B$  is the backward shift operator; that is,  $BZ_t = Z_{t-1}$ . Let  $d$  denote the maximum of the orders of differencing,  $d_i$ , over the  $k$  series. The routine computes values of the differenced/transformed series  $W_t = (w_{1t}, w_{2t}, \dots, w_{kt})^T$  for  $t = d + 1, d + 2, \dots, n$  as follows:

$$w_{it} = \delta_i(B)z_{it}^*, \quad i = 1, 2, \dots, k$$

where  $z_{it}^*$  are the transformed values of the original  $k$ -dimensional time series  $Z_t = (z_{1t}, z_{2t}, \dots, z_{kt})^T$ .

The differencing parameters  $\delta_{ij}$ , for  $i = 1, 2, \dots, k$ ;  $j = 1, 2, \dots, d_i$  must be supplied by the user. If the  $i$ th series does not require differencing, then  $d_i = 0$ .

#### 4 References

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

Wei W W S (1990) *Time Series Analysis: Univariate and Multivariate Methods* Addison-Wesley

#### 5 Parameters

- 1: **k** – Integer *Input*  
*On entry:* the dimension,  $k$ , of the multivariate time series.  
*Constraint:*  $k \geq 1$ .
- 2: **n** – Integer *Input*  
*On entry:* the number,  $n$ , of observations in the series, prior to differencing.  
*Constraint:*  $n \geq 1$ .
- 3: **z[dim]** – double *Input*  
**Note:** the dimension,  $dim$ , of the array **z** must be at least  $k \times n$ .  
*On entry:* **z**[( $t - 1$ ) $k + i - 1$ ] must contain the  $i$ th series at time  $t$ , for  $t = 1, 2, \dots, n$ ;  $i = 1, 2, \dots, k$ .

- 4: **tr**[**k**] – Integer *Input*  
*On entry:* **tr**[ $i - 1$ ] indicates whether the  $i$ th series is to be transformed, for  $i = 1, 2, \dots, k$ .  
 If **tr**[ $i$ ] = -1, a square root transformation is used.  
 If **tr**[ $i$ ] = 0, no transformation is used.  
 If **tr**[ $i$ ] = 1, a log transformation is used.  
*Constraint:* **tr**[ $i$ ] = -1, 0 or 1 for  $i = 0, 1, \dots, k - 1$ .
- 5: **id**[**k**] – Integer *Input*  
*On entry:* the order of differencing for each series,  $d_1, d_2, \dots, d_k$  with  $d = \max(\mathbf{id}[i])$ , for  $i = 0, 1, \dots, k - 1$ .  
*Constraint:*  $0 \leq \mathbf{id}[i] < \mathbf{n}$  for  $i = 0, 1, \dots, k - 1$ .
- 6: **delta**[*dim*] – double *Input*  
**Note:** the dimension, *dim*, of the array **delta** must be at least  $\max(1, k \times d)$ .  
*On entry:* if **id**[ $i - 1$ ] > 0 then **delta**[( $j - 1$ ) $k + i - 1$ ] must be set to  $\delta_{ij}$ , for  $j = 1, 2, \dots, d_i$ ;  $i = 1, 2, \dots, k$ .
- 7: **w**[*dim*] – double *Output*  
**Note:** the dimension, *dim*, of the array **w** must be at least  $k \times (n - d)$ .  
*On exit:* **w**[( $t - 1$ ) $k + i - 1$ ] contains the value of  $w_{i,t+d}$ , for  $i = 1, 2, \dots, k$ ;  $t = 1, 2, \dots, n - d$ .
- 8: **nd** – Integer \* *Output*  
*On exit:* the number of differenced values,  $n - d$ , in the series.
- 9: **fail** – NagError \* *Input/Output*  
 The NAG error parameter (see the Essential Introduction).

## 6 Error Indicators and Warnings

### NE\_INT

On entry, **k** = *<value>*.  
 Constraint: **k**  $\geq 1$ .

On entry, **n** = *<value>*.  
 Constraint: **n**  $\geq 1$ .

### NE\_INT\_ARRAY

On entry, **tr**[ $i$ ]**id**[ $i$ ] = *<value>*.  
 Constraint: **tr**[ $i$ ] = -1, 0 or 1 for  $i = 0, \dots, k - 1$ .

On entry, **tr**[ $i$ ]**id**[ $i$ ] = *<value>*.  
 Constraint:  $0 \leq \mathbf{id}[i] < \mathbf{n}$  for  $i = 0, \dots, k - 1$ .

### NE\_INT\_ARRAY\_ELEM\_CONS

On entry, element *<value>* of **id** is greater than or equal to **n**.

On entry, element *<value>* of **id** is less than zero.

### NE\_TRANSFORMATION

On entry, one (or more) of the transformations requested is invalid.

**NE\_ALLOC\_FAIL**

Memory allocation failed.

**NE\_BAD\_PARAM**

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

**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.

**7 Accuracy**

The computations are believed to be stable.

**8 Further Comments**

The same differencing operator does not have to be applied to all the series. For example, suppose we have  $k = 2$ , and wish to apply the second-order differencing operator  $\nabla^2$  to the first series and the first-order differencing operator  $\nabla$  to the second series:

$$w_{1t} = \nabla^2 z_{1t} = (1 - B)^2 z_{1t} = (1 - 2B + B^2) z_{1t}, \quad \text{and}$$

$$w_{2t} = \nabla z_{2t} = (1 - B) z_{2t}.$$

Then  $d_1 = 2, d_2 = 1, d = \max(d_1, d_2) = 2$ , and

$$\mathbf{delta} = \begin{bmatrix} \delta_{11} & \delta_{12} \\ \delta_{21} & \delta_{22} \end{bmatrix} = \begin{bmatrix} 2 & -1 \\ 1 & 1 \end{bmatrix}.$$

**9 Example**

A program to difference (nonseasonally) each of two time series of length 48. No transformation is to be applied to either of the series.

**9.1 Program Text**

```

/* nag_tsa_multi_diff (g13dlc) Example Program.
 *
 * Copyright 2002 Numerical Algorithms Group.
 *
 * Mark 7, 2002.
 */

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

int main(void)
{
    /* Scalars */
    Integer exit_status, i, j, k, maxd, mind, n, nd, ik, nw,
        pdw, pddelta, pdz;
    NagError fail;

    /* Arrays */
    double *delta = 0, *w = 0, *z = 0;
    Integer *id = 0, *tr=0;

#define W(I,J) w[(J-1)*pdw + I - 1]
#define DELTA(I,J) delta[(J-1)*pddelta + I - 1]
#define Z(I,J) z[(J-1)*pdz + I - 1]

```

```

INIT_FAIL(fail);
exit_status = 0;

Vprintf("g13dlc Example Program Results\n");

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

Vscanf("%ld%ld%*[\n] ", &k, &n);

if (k > 0 && n > 0)
{
    ik = k;
    /* Allocate array id */
    if ( !(id = NAG_ALLOC(k, Integer)) )
    {
        Vprintf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }

    for (i = 1; i <= k; ++i)
        Vscanf("%ld", &id[i-1]);
    Vscanf("%*[\n] ");

    mind = 0;
    maxd = 0;
    for (i = 1; i <= k; ++i)
    {
        mind = MIN(mind, id[i-1]);
        maxd = MAX(maxd, id[i-1]);
    }

    if (mind >= 0)
    {
        /* Allocate arrays */
        nw = n - maxd;
        if ( !(tr = NAG_ALLOC(k, Integer)) ||
            !(delta = NAG_ALLOC(ik * maxd, double)) ||
            !(w = NAG_ALLOC(ik * nw, double)) ||
            !(z = NAG_ALLOC(ik * n, double)) )
        {
            Vprintf("Allocation failure\n");
            exit_status = -1;
            goto END;
        }

        pdw = ik;
        pddelta = ik;
        pdz = ik;

        for (i = 1; i <= k; ++i)
        {
            for (j = 1; j <= n; ++j)
                Vscanf("%lf", &Z(i,j));
            Vscanf("%*[\n] ");
        }

        for (i = 1; i <= k; ++i)
            Vscanf("%ld", &tr[i-1]);
        Vscanf("%*[\n] ");

        if (maxd > 0)
        {
            for (i = 1; i <= k; ++i)
            {
                for (j = 1; j <= id[i-1]; ++j)
                    Vscanf("%lf", &DELTA(i,j));
                Vscanf("%*[\n] ");
            }
        }
    }
}

```

```

    }

    g13dlc(k, n, z, tr, id, delta, w, &nd, &fail);
    if (fail.code != NE_NOERROR)
    {
        Vprintf("Error from g13dlc.\n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }
    Vprintf("\n");
    Vprintf(" Transformed/Differenced series\n");
    Vprintf(" ----- \n");

    for (i = 1; i <= k; ++i)
    {
        Vprintf("\n");
        Vprintf(" Series %2ld\n", i);
        Vprintf(" ----- \n");
        Vprintf("\n");
        Vprintf(" Number of differenced values = %6ld\n", nd);
        Vprintf("\n");
        for (j = 1; j <= nd; ++j)
        {
            Vprintf("%10.3f", W(i,j));
            if (j % 8 == 0 || j == nd)
                Vprintf("\n");
        }
    }
}

END:
if (tr) NAG_FREE(tr);
if (delta) NAG_FREE(delta);
if (w) NAG_FREE(w);
if (z) NAG_FREE(z);
if (id) NAG_FREE(id);

return exit_status;
}

```

## 9.2 Program Data

```

g13dlc Example Program Data
 2 48 : k, n
 1 1 : id(0), id(1)
-1.490 -1.620  5.200  6.230  6.210  5.860  4.090  3.180
 2.620  1.490  1.170  0.850 -0.350  0.240  2.440  2.580
 2.040  0.400  2.260  3.340  5.090  5.000  4.780  4.110
 3.450  1.650  1.290  4.090  6.320  7.500  3.890  1.580
 5.210  5.250  4.930  7.380  5.870  5.810  9.680  9.070
 7.290  7.840  7.550  7.320  7.970  7.760  7.000  8.350
 7.340  6.350  6.960  8.540  6.620  4.970  4.550  4.810
 4.750  4.760 10.880 10.010 11.620 10.360  6.400  6.240
 7.930  4.040  3.730  5.600  5.350  6.810  8.270  7.680
 6.650  6.080 10.250  9.140 17.750 13.300  9.630  6.800
 4.080  5.060  4.940  6.650  7.940 10.760 11.890  5.850
 9.010  7.500 10.020 10.380  8.150  8.370 10.730 12.140 : End of time series
 0 0 : tr(0), tr(1)
 1.0 : delta(1,1)
 1.0 : delta(2,1)

```

## 9.3 Program Results

g13dlc Example Program Results

```

Transformed/Differenced series
-----

```

```

Series 1

```

-----

Number of differenced values = 47

-0.130	6.820	1.030	-0.020	-0.350	-1.770	-0.910	-0.560
-1.130	-0.320	-0.320	-1.200	0.590	2.200	0.140	-0.540
-1.640	1.860	1.080	1.750	-0.090	-0.220	-0.670	-0.660
-1.800	-0.360	2.800	2.230	1.180	-3.610	-2.310	3.630
0.040	-0.320	2.450	-1.510	-0.060	3.870	-0.610	-1.780
0.550	-0.290	-0.230	0.650	-0.210	-0.760	1.350	

Series 2

-----

Number of differenced values = 47

-0.990	0.610	1.580	-1.920	-1.650	-0.420	0.260	-0.060
0.010	6.120	-0.870	1.610	-1.260	-3.960	-0.160	1.690
-3.890	-0.310	1.870	-0.250	1.460	1.460	-0.590	-1.030
-0.570	4.170	-1.110	8.610	-4.450	-3.670	-2.830	-2.720
0.980	-0.120	1.710	1.290	2.820	1.130	-6.040	3.160
-1.510	2.520	0.360	-2.230	0.220	2.360	1.410	

---