NAG Library Function Document

\texttt{nag_tsa_multi_diff (g13dlc)}

1 Purpose

\texttt{nag_tsa_multi_diff (g13dlc)} differences and/or transforms a multivariate time series.

2 Specification

\begin{verbatim}
#include <nag.h>
#include <nagg13.h>

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)
\end{verbatim}

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 function also allows you 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 \(\delta_i(B) = 1 - \delta_1 B - \delta_2 B^2 - \cdots - \delta_{d_i} B^{d_i}\), 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 function computes values of the differenced/transformed series \(W_t = (w_{1t}, w_{2t}, \ldots, w_{kt})^T\), for \(t = d + 1, \ldots, n\), as follows:

\[ w_{it} = \delta_i(B)z_{it}, \quad i = 1, 2, \ldots, k \]

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

The differencing parameters \(\delta_{ij}\), for \(i = 1, 2, \ldots, k\) and \(j = 1, 2, \ldots, d_i\), must be supplied by you. If the \(i\)th series does not require differencing, then \(d_i = 0\).

4 References


5 Arguments

1: \(k\) – Integer \hspace{1cm} Input

\textit{On entry}: \(k\), the dimension of the multivariate time series.

\textit{Constraint}: \(k \geq 1\).

2: \(n\) – Integer \hspace{1cm} Input

\textit{On entry}: \(n\), the number of observations in the series, prior to differencing.

\textit{Constraint}: \(n \geq 1\).

3: \(z[k \times n]\) – const double \hspace{1cm} Input

\textit{On entry}: \(z(t-1)k + i - 1\) must contain the \(i\)th series at time \(t\), for \(t = 1, 2, \ldots, n\) and \(i = 1, 2, \ldots, k\).
4: \( \text{tr}[k] \) – const Integer

*Input*

*On entry:* \( \text{tr}[i-1] \) indicates whether the \( i \)th series is to be transformed, for \( i = 1, 2, \ldots, k \).

\( \text{tr}[i-1] = -1 \)

A square root transformation is used.

\( \text{tr}[i-1] = 0 \)

No transformation is used.

\( \text{tr}[i-1] = 1 \)

A log transformation is used.

*Constraint:* \( \text{tr}[i-1] = -1, 0 \) or \( 1 \), for \( i = 1, 2, \ldots, k \).

5: \( \text{id}[k] \) – const Integer

*Input*

*On entry:* the order of differencing for each series, \( d_1, d_2, \ldots, d_k \).

*Constraint:* \( 0 \leq \text{id}[i] < n \), for \( i = 0, 1, \ldots, k-1 \).

6: \( \text{delta}[\text{dim}] \) – const double

*Input*

*Note:* the dimension, \( \text{dim} \), of the array \( \text{delta} \) must be at least \( k \times \max(1, d) \), where \( d = \max(\text{id}[i-1]) \).

*On entry:* if \( \text{id}[i-1] > 0 \) then \( \text{delta}[(j-1)k + i-1] \) must be set to \( \delta_{ij} \), for \( j = 1, 2, \ldots, d_i \) and \( i = 1, 2, \ldots, k \).

7: \( \text{w}[\text{dim}] \) – double

*Output*

*Note:* the dimension, \( \text{dim} \), of the array \( \text{w} \) must be at least \( k \times (n - d) \), where \( d = \max(\text{id}[i-1]) \).

*On exit:* \( \text{w}[(t-1)k + i-1] \) contains the value of \( w_{i,t+d} \), for \( i = 1, 2, \ldots, k \) and \( t = 1, 2, \ldots, n - d \).

8: \( \text{nd} \) – Integer

*Output*

*On exit:* the number of differenced values, \( n - d \), in the series, where \( d = \max(\text{id}[i-1]) \).

9: \( \text{fail} \) – NagError

*Input/Output*

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

6 **Error Indicators and Warnings**

**NE_ALLOC_FAIL**

Dynamic memory allocation failed.

See Section 3.2.1.2 in the Essential Introduction for further information.

**NE_BAD_PARAM**

On entry, argument \( \langle \text{value} \rangle \) had an illegal value.

**NE_INT**

On entry, \( k = \langle \text{value} \rangle \).

Constraint: \( k \geq 1 \).

On entry, \( n = \langle \text{value} \rangle \).

Constraint: \( n \geq 1 \).

**NE_INT_ARRAY**

On entry, element \( \langle \text{value} \rangle \) of \( \text{id} \) is greater than or equal to \( n \).
On entry, element \(\text{value}\) of \text{id} is less than zero.
On entry, \(\text{tr}[\text{value}] = \text{value}\).
Constraint: \(\text{tr}[i] = -1, 0\) or \(1\).

**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 3.6.6 in the Essential Introduction for further information.

**NE_NO_LICENCE**
Your licence key may have expired or may not have been installed correctly.
See Section 3.6.5 in the Essential Introduction for further information.

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

7 Accuracy
The computations are believed to be stable.

8 Parallelism and Performance
nag_tsa_multi_diff (g13dlc) is not threaded by NAG in any implementation.

nag_tsa_multi_diff (g13dlc) 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 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
\[
\delta = \begin{bmatrix} \delta_{11} & \delta_{12} \\ \delta_{21} & \delta_{22} \end{bmatrix} = \begin{bmatrix} 2 & -1 \\ 1 & -1 \end{bmatrix}.
\]

10 Example
A program to difference (non-seasonally) each of two time series of length 48. No transformation is to be applied to either of the series.
10.1 Program Text

/* nag_tsa_multi_diff (g13dlc) Example Program. */
* Copyright 2014 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, nw,
            pdw, pddelta, kmax;
    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)*kmax + I - 1]

    INIT_FAIL(fail);
    exit_status = 0;
    printf("nag_tsa_multi_diff (g13dlc) Example Program Results\n");

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

    if( k>0 && n>0 )
    {
        kmax = k;
        /* Allocate array id */
        if (!((id = NAG_ALLOC(k, Integer)))
        {
            printf("Allocation failure\n");
            exit_status = -1;
            goto END;
        }
        for (i = 1; i <= k; ++i)
        {
            ifdef _WIN32
                scanf_s("%"NAG_IFMT"%"NAG_IFMT"%*[\n] ", &k, &n);
            else
                scanf("%"NAG_IFMT"%"NAG_IFMT"%*[\n] ", &k, &n);
        #endif

        #ifdef _WIN32
            scanf_s("%"NAG_IFMT"", &id[i-1]);
        #else
            scanf("%"NAG_IFMT"", &id[i-1]);
        #endif
        #ifdef _WIN32
            scanf("%*[\n] ");
        #else
            scanf("%*[\n] ");
        #endif

        #ifdef _WIN32
            scanf_s("%"NAG_IFMT"", &id[i-1]);
        #else
            scanf("%"NAG_IFMT"", &id[i-1]);
        #endif
        #ifdef _WIN32
            scanf("%*[\n] ");
        #else
            scanf("%*[\n] ");
        #endif

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

        ifdef _WIN32
            scanf_s("%"NAG_IFMT"", &id[i-1]);
        else
            scanf("%"NAG_IFMT"", &id[i-1]);
        endif
        #ifdef _WIN32
            scanf("%*[\n] ");
        else
            scanf("%*[\n] ");
        #endif
    endif
}

END:
exit_status = fail.text;
printf("nag_tsa_multi_diff (g13dlc) Example Program Results\n");
    return exit_status;
}

END:
exit_status = fail.text;
printf("nag_tsa_multi_diff (g13dlc) Example Program Results\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(kmax * maxd, double))
        || !(w = NAG_ALLOC(kmax * nw, double))
        || !(z = NAG_ALLOC(kmax * n, double)))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }

    pdw = kmax;
pdedelta = kmax;
    for (i = 1; i <= k; ++i)
    {
        for (j = 1; j <= n; ++j)
            scanf("%lf", &Z(i, j));
        scanf("%*[\n] ");
    }
    for (i = 1; i <= k; ++i)
    {
        scanf("%"NAG_IFMT", &tr[i-1]);
    }
    if (maxd > 0)
    {
        for (i = 1; i <= k; ++i)
        {
            for (j = 1; j <= id[i-1]; ++j)
                scanf("%lf", &DELT(A(i, j));
        }
    }
    /* nag_tsa_multi_diff (g13dlc).
* Multivariate time series, differences and/or transforms */

nag_tsa_multi_diff(k, n, z, tr, id, delta, w, &nd, &fail);

if (fail.code != NE_NOERROR)
{
    printf("Error from nag_tsa_multi_diff (g13dlc).\n%\s\n", fail.message);
    exit_status = 1;
    goto END;
}
printf("\n");
printf(" Transformed/Differenced series\n");
printf(" ------------------------------\n");
for (i = 1; i <= k; ++i)
{
    printf("\n");
    printf(" Series %2"NAG_IFMT"\n", i);
    printf(" -----------\n");
    printf("\n");
    printf(" Number of differenced values = %6"NAG_IFMT"\n", nd);
    printf("\n");
    printf("\n");
    for (j = 1; j <= nd; ++j)
    {
        printf("%10.3f", W(i, j));
        if (j % 8 == 0 || j == nd)
            printf("\n");
    }
}
}
}

END:
NAG_FREE(tr);
NAG_FREE(delta);
NAG_FREE(w);
NAG_FREE(z);
NAG_FREE(id);
return exit_status;

10.2 Program Data

nag_tsa_multi_diff (g13dlc) Example Program Data
2 48 : k, n
1 1 : id(0), id(1)
2.620 1.490 1.170 0.850 -0.350 0.240 2.440 2.580
2.040 2.260 3.340 5.090 5.000 4.780 4.110
7.290 7.840 7.550 7.320 7.970 7.760 7.000 8.350
4.080 5.060 4.940 6.650 7.940 10.760 11.890 5.850
0 0 : tr(0), tr(1)
1.0 : delta(1,1)
1.0 : delta(2,1)
10.3 Program Results

nag_tsa_multi_diff (g13dlc) Example Program Results

Transformed/Differenced series
-----------------------------

Series 1
-----------

Number of differenced values = 47

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

Series 2
-----------

Number of differenced values = 47

\[ \begin{align*}
&-0.990 & 0.610 & 1.580 & -1.920 & -1.650 & -0.420 & 0.260 & -0.060 \\
&0.010 & 6.120 & -0.870 & 1.610 & -1.260 & -3.960 & -0.160 & 1.690 \\
&-3.890 & -0.310 & 1.870 & -0.250 & 1.460 & 1.460 & -0.590 & -1.030 \\
&0.980 & -0.120 & 1.710 & 1.290 & 2.820 & 1.130 & -6.040 & 3.160 \\
&-1.510 & 2.520 & 0.360 & -2.230 & 0.220 & 2.360 & 1.410 \\
\end{align*} \]