NAG Library Function Document

nag_tsa_diff (g13aac)

1 Purpose

nag_tsa_diff (g13aac) carries out non-seasonal and seasonal differencing on a time series. Information which allows the original series to be reconstituted from the differenced series is also produced. This information is required in time series forecasting.

2 Specification

```
#include <nag.h>
#include <nagg13.h>
void nag_tsa_diff (const double x[], Integer nx, Integer d, Integer ds,
     Integer s, double xd[], Integer *nxd, NagError *fail)
```

3 Description

Let $\nabla^d \nabla_s^D x_i$ be the ith value of a time series x_i, for $i = 1, 2, \ldots, n$ after non-seasonal differencing of order d and seasonal differencing of order D (with period or seasonality s). In general,

$$\begin{aligned} \nabla^d \nabla_s^D x_i &= \nabla^{d-1} \nabla_s^D x_{i+1} - \nabla^{d-1} \nabla_s^D x_i && d > 0 \\ \nabla^d \nabla_s^D x_i &= \nabla^d \nabla_s^{D-1} x_{i+s} - \nabla^d \nabla_s^{D-1} x_i && D > 0 \end{aligned}$$

Non-seasonal differencing up to the required order d is obtained using

$$\begin{aligned} \nabla^1 x_i &= x_{i+1} - x_i && \text{for } i = 1, 2, \ldots, (n-1) \\ \nabla^2 x_i &= \nabla^1 x_{i+1} - \nabla^1 x_i && \text{for } i = 1, 2, \ldots, (n-2) \\ &\vdots \\ \nabla^d x_i &= \nabla^{d-1} x_{i+1} - \nabla^{d-1} x_i && \text{for } i = 1, 2, \ldots, (n-d) \end{aligned}$$

Seasonal differencing up to the required order D is then obtained using

$$\begin{aligned} \nabla^d \nabla_s^1 x_i &= \nabla^d x_{i+s} - \nabla^d x_i && \text{for } i = 1, 2, \ldots, (n-d-s) \\ \nabla^d \nabla_s^2 x_i &= \nabla^d \nabla_s^1 x_{i+s} - \nabla^d \nabla_s^1 x_i && \text{for } i = 1, 2, \ldots, (n-d-2s) \\ &\vdots \\ \nabla^d \nabla_s^D x_i &= \nabla^d \nabla_s^{D-1} x_{i+s} - \nabla^d \nabla_s^{D-1} x_i && \text{for } i = 1, 2, \ldots, (n-d-D \times s) \end{aligned}$$

Mathematically, the sequence in which the differencing operations are performed does not affect the final resulting series of $m = n - d - D \times s$ values.

4 References

None.

5 Arguments

1: **x**[**nx**] – const double *Input*

On entry: the undifferenced time series, x_i, for $i = 1, 2, \ldots, n$.

2: **nx** – Integer *Input*

On entry: n, the number of values in the undifferenced time series.

Constraint: $\mathbf{nx} > \mathbf{d} + (\mathbf{ds} \times \mathbf{s})$.

3: **d** – Integer *Input*

On entry: d, the order of non-seasonal differencing.

Constraint: $\mathbf{d} \geq 0$.

4: **ds** – Integer *Input*

On entry: D, the order of seasonal differencing.

Constraint: $\mathbf{ds} \geq 0$.

5: **s** – Integer *Input*

On entry: s, the seasonality.

Constraints:

if $\mathbf{ds} > 0$, $\mathbf{s} > 0$;
if $\mathbf{ds} = 0$, $\mathbf{s} \geq 0$.

6: **xd**[**nx**] – double *Output*

On exit: the differenced values in elements 0 to $\mathbf{nxd} - 1$, and reconstitution data in the remainder of the array.

7: **nxd** – Integer * *Output*

On exit: the number of differenced values in the array **xd**.

8: **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 ⟨*value*⟩ had an illegal value.

NE_INT

On entry, $\mathbf{d} = \langle value \rangle$.
Constraint: $\mathbf{d} \geq 0$.

On entry, $\mathbf{ds} = \langle value \rangle$.
Constraint: $\mathbf{ds} \geq 0$.

On entry, $\mathbf{s} = \langle value \rangle$.
Constraint: $\mathbf{s} \geq 0$.

NE_INT_2

On entry, $\mathbf{ds} = \langle value \rangle$.
Constraint: if $\mathbf{s} = 0$, $\mathbf{ds} \leq 0$.

NE_INT_4

On entry, **nx** = ⟨*value*⟩, **d** = ⟨*value*⟩, **ds** = ⟨*value*⟩ and **s** = ⟨*value*⟩.
Constraint: **nx** > **d** + (**ds** × **s**).

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.

7 Accuracy

The computations are believed to be stable.

8 Parallelism and Performance

Not applicable.

9 Further Comments

The time taken by nag_tsa_diff (g13aac) is approximately proportional to (**d** + **ds**) × **nx**.

10 Example

This example reads in a set of data consisting of 20 observations from a time series. Non-seasonal differencing of order 2 and seasonal differencing of order 1 (with seasonality of 4) are applied to the input data, giving an output array holding 14 differenced values and 6 values which can be used to reconstitute the output array.

10.1 Program Text

```
/* nag_tsa_diff (g13aac) 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, d, ds, s, nx, nxd;
  NagError fail;

  /* Arrays */
  double   *x = 0, *xd = 0;

  INIT_FAIL(fail);

  exit_status = 0;
  printf("nag_tsa_diff (g13aac) Example Program Results\n");
```

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

#ifdef _WIN32
  scanf_s("%"NAG_IFMT"%"NAG_IFMT"%"NAG_IFMT"%"NAG_IFMT"%*[^\n] ", &nx, &d,
          &ds, &s);
#else
  scanf("%"NAG_IFMT"%"NAG_IFMT"%"NAG_IFMT"%"NAG_IFMT"%*[^\n] ", &nx, &d,
          &ds, &s);
#endif

  if (nx > 0)
    {
      /* Allocate memory */
      if (!(x = NAG_ALLOC(nx, double)) ||
          !(xd = NAG_ALLOC(nx, double)))
        {
          printf("Allocation failure\n");
          exit_status = -1;
          goto END;
        }

      for (i = 1; i <= nx; ++i)
#ifdef _WIN32
        scanf_s("%lf", &x[i-1]);
#else
        scanf("%lf", &x[i-1]);
#endif
#ifdef _WIN32
      scanf_s("%*[^\n] ");
#else
      scanf("%*[^\n] ");
#endif

      printf("\n");
      printf("Non-seasonal differencing of order %"NAG_IFMT" "
             "and seasonal differencing\nof order %"NAG_IFMT" "
             "with seasonality %"NAG_IFMT" are applied\n", d, ds, s);

      /* nag_tsa_diff (g13aac).
       * Univariate time series, seasonal and non-seasonal
       * differencing
       */
      nag_tsa_diff(x, nx, d, ds, s, xd, &nxd, &fail);
      if (fail.code != NE_NOERROR)
        {
          printf("Error from nag_tsa_diff (g13aac).\n%s\n",
                  fail.message);
          exit_status = 1;
          goto END;
        }

      printf("\n");
      printf("The output array holds %2"NAG_IFMT" values, of which the "
             "first %2"NAG_IFMT" are differenced values\n\n", nx, nxd);

      for (i = 1; i <= nx; ++i)
        {
          printf("%10.1f", xd[i-1]);
          if (i % 5 == 0 || i == nx)
            printf("\n");
        }
    }

 END:
```

```
  NAG_FREE(x);
  NAG_FREE(xd);

  return exit_status;
}
```

10.2 Program Data

```
nag_tsa_diff (g13aac) Example Program Data
 20  2  1  4
 120.0 108.0  98.0 118.0 135.0
 131.0 118.0 125.0 121.0 100.0
  82.0  82.0  89.0  88.0  86.0
  96.0 108.0 110.0  99.0 105.0
```

10.3 Program Results

```
nag_tsa_diff (g13aac) Example Program Results

Non-seasonal differencing of order 2 and seasonal differencing
of order 1 with seasonality 4 are applied

The output array holds 20 values, of which the first 14 are differenced values

     -11.0     -10.0      -8.0       4.0      12.0
      -2.0      18.0       9.0      -4.0      -6.0
      -5.0      -2.0     -12.0       5.0       2.0
     -10.0     -13.0      17.0       6.0     105.0
```