NAG Library Function Document

nag_tsa_spectrum_bivar (g13cdc)

1 Purpose

nag_tsa_spectrum_bivar (g13cdc) calculates the smoothed sample cross spectrum of a bivariate time series using spectral smoothing by the trapezium frequency (Daniell) window.

2 Specification

```
#include <nag.h>
#include <nagg13.h>
void nag_tsa_spectrum_bivar (Integer nxy, NagMeanOrTrend mt_correction,
     double pxy, Integer mw, Integer is, double pw, Integer l, Integer kc,
     const double x[], const double y[], Complex **g, Integer *ng,
     NagError *fail)
```

3 Description

The supplied time series may be mean and trend corrected and tapered as in the description of nag_tsa_spectrum_univar (g13cbc) before calculation of the unsmoothed sample cross-spectrum

$$f_{xy}^*(\omega) = \frac{1}{2\pi n}\left\{\sum_{t=1}^{n} y_t \exp(i\omega t)\right\} \times \left\{\sum_{t=1}^{n} x_t \exp(-i\omega t)\right\}$$

for frequency values $\omega_j = \frac{2\pi j}{K}$, $0 \leq \omega_j \leq \pi$.

A correction is made for bias due to any tapering.

As in the description of nag_tsa_spectrum_univar (g13cbc) for univariate frequency window smoothing, the smoothed spectrum is returned at a subset of these frequencies,

$$\nu_l = \frac{2\pi l}{L}, l = 0, 1, \ldots, [L/2]$$

where [ ] denotes the integer part.

Its real part or co-spectrum $cf(\nu_l)$, and imaginary part or quadrature spectrum $qf(\nu_l)$ are defined by

$$f_{xy}(\nu_l) = cf(\nu_l) + iqf(\nu_l) = \sum_{|\omega_k| < \pi/M} \tilde{w}_k f_{xy}^*(\nu_l + \omega_k)$$

where the weights $\tilde{w}_k$ are similar to the weights w_k defined for nag_tsa_spectrum_univar (g13cbc), but allow for an implicit alignment shift S between the series:

$$\tilde{w}_k = w_k \exp(-2\pi i S k/L).$$

It is recommended that S is chosen as the lag k at which the cross-covariances $c_{xy}(k)$ peak, so as to minimize bias.

If no smoothing is required, the integer M which determines the frequency window width $\frac{2\pi}{M}$, should be set to n.

The bandwidth of the estimates will normally have been calculated in a previous call of nag_tsa_spectrum_univar (g13cbc) for estimating the univariate spectra of y_t and x_t.

4 References

Bloomfield P (1976) *Fourier Analysis of Time Series: An Introduction* Wiley

Jenkins G M and Watts D G (1968) *Spectral Analysis and its Applications* Holden–Day

5 Arguments

1: **nxy** – Integer *Input*

On entry: the length of the time series x and y, n.

Constraint: **nxy** ≥ 1.

2: **mt_correction** – NagMeanOrTrend *Input*

On entry: whether the data are to be initially mean or trend corrected. **mt_correction** = Nag_NoCorrection for no correction, **mt_correction** = Nag_Mean for mean correction, **mt_correction** = Nag_Trend for trend correction.

Constraint: **mt_correction** = Nag_NoCorrection, Nag_Mean or Nag_Trend.

3: **pxy** – double *Input*

On entry: the proportion of the data (totalled over both ends) to be initially tapered by the split cosine bell taper.

A value of 0.0 implies no tapering.

Constraint: $0.0 \leq$ **pxy** ≤ 1.0.

4: **mw** – Integer *Input*

On entry: the frequency width, M, of the smoothing window as $2\pi/M$.

A value of n implies that no smoothing is to be carried out.

Constraint: $1 \leq$ **mw** $\leq$ **nxy**.

5: **is** – Integer *Input*

On entry: the alignment shift, S, between the x and y series. If x leads y, the shift is positive.

Constraint: $-$**l** $<$ **is** $<$ **l**.

6: **pw** – double *Input*

On entry: the shape argument, p, of the trapezium frequency window.

A value of 0.0 gives a triangular window, and a value of 1.0 a rectangular window.

If **mw** = **nxy** (i.e., no smoothing is carried out) then **pw** is not used.

Constraint: $0.0 \leq$ **pw** ≤ 1.0 if **mw** $\neq$ **nxy**.

7: **l** – Integer *Input*

On entry: the frequency division, L, of smoothed cross spectral estimates as $2\pi/L$.

Constraint: **l** ≥ 1.

l must be a factor of **kc** (see below).

8: **kc** – Integer *Input*

On entry: the order of the fast Fourier transform (FFT) used to calculate the spectral estimates. **kc** should be a product of small primes such as 2^m where m is the smallest integer such that $2^m \geq 2n$, provided $m \leq 20$.

Constraints:

kc $\geq 2 \times$ **nxy**;
kc must be a multiple of **l**. The largest prime factor of **kc** must not exceed 19, and the total number of prime factors of **kc**, counting repetitions, must not exceed 20. These two restrictions are imposed by the internal FFT algorithm used.

9: **x**[**kc**] – const double *Input*

On entry: the **nxy** data points of the x series.

10: **y**[**kc**] – const double *Input*

On entry: the **nxy** data points of the y series.

11: **g** – Complex ** *Output*

On exit: the complex vector which contains the **ng** cross spectral estimates in elements **g**[0] to **g**[**ng** − 1]. The y series leads the x series.

The memory for this vector is allocated internally. If no memory is allocated to **g** (e.g., when an input error is detected) then **g** will be **NULL** on return. If repeated calls to this function are required then `NAG_FREE` should be used to free the memory in between calls.

12: **ng** – Integer * *Output*

On exit: the number of spectral estimates, $[L/2] + 1$, whose separate parts are held in **g**.

13: **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_2_INT_ARG_CONS

On entry, **kc** = ⟨*value*⟩ while **l** = ⟨*value*⟩. These arguments must satisfy **kc**%**l** ≠ 0 when **l** > 0.

On entry, **kc** = ⟨*value*⟩ while **nxy** = ⟨*value*⟩. These arguments must satisfy **kc** ≥ 2*****nxy** when **nxy** > 0.

On entry, **l** = ⟨*value*⟩ while **is** = ⟨*value*⟩. These arguments must satisfy ||**is**|| < **l** when **l** > 0.

NE_2_INT_ARG_GT

On entry, **mw** = ⟨*value*⟩ while **nxy** = ⟨*value*⟩. These arguments must satisfy **mw** ≤ **nxy**.

NE_ALLOC_FAIL

Dynamic memory allocation failed.

NE_BAD_PARAM

On entry, argument **mt_correction** had an illegal value.

NE_FACTOR_GT

At least one of the prime factors of **kc** is greater than 19.

NE_INT_ARG_LT

On entry, **l** = ⟨*value*⟩.
Constraint: **l** ≥ 1.

On entry, **mw** $= \langle value \rangle$.
Constraint: **mw** ≥ 1.

On entry, **nxy** $= \langle value \rangle$.
Constraint: **nxy** ≥ 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.

NE_REAL_ARG_GT

On entry, **pw** must not be greater than 1.0: **pw** $= \langle value \rangle$.

On entry, **pxy** must not be greater than 1.0: **pxy** $= \langle value \rangle$.

NE_REAL_ARG_LT

On entry, **pw** must not be less than 0.0: **pw** $= \langle value \rangle$.

On entry, **pxy** must not be less than 0.0: **pxy** $= \langle value \rangle$.

NE_TOO_MANY_FACTORS

kc has more than 20 prime factors.

7 Accuracy

The FFT is a numerically stable process, and any errors introduced during the computation will normally be insignificant compared with uncertainty in the data.

8 Parallelism and Performance

nag_tsa_spectrum_bivar (g13cdc) is not threaded in any implementation.

9 Further Comments

nag_tsa_spectrum_bivar (g13cdc) carries out an FFT of length **kc** to calculate the sample cross spectrum. The time taken by the function for this is approximately proportional to **kc** $\times \log(\mathbf{kc})$ (but see function document nag_sum_fft_realherm_1d (c06pac) for further details).

10 Example

The example program reads 2 time series of length 296. It selects mean correction and a 10% tapering proportion. It selects a $2\pi/16$ frequency width of smoothing window, a window shape argument of 0.5 and an alignment shift of 3. It then calls nag_tsa_spectrum_bivar (g13cdc) to calculate the smoothed sample cross spectrum and prints the results.

10.1 Program Text

```
/* nag_tsa_spectrum_bivar (g13cdc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 *
 */

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

```
#include <nag_stdlib.h>
#include <naga02.h>
#include <nagg13.h>

#define L      80
#define KC     8*L
#define NXYMAX 300

int main(void)
{

  Complex *g;
  Integer exit_status = 0, i, is, j, kc = KC, l = L, mw, ng, nxy;
  NagError fail;
  double pw, pxy, *x = 0, *y = 0;

  INIT_FAIL(fail);

  printf("nag_tsa_spectrum_bivar (g13cdc) Example Program Results\n");

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

#ifdef _WIN32
  scanf_s("%" NAG_IFMT " ", &nxy);
#else
  scanf("%" NAG_IFMT " ", &nxy);
#endif
  if (nxy > 0 && nxy <= NXYMAX) {
    if (!(x = NAG_ALLOC(KC, double)) || !(y = NAG_ALLOC(KC, double)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }
    for (i = 1; i <= nxy; ++i)
#ifdef _WIN32
      scanf_s("%lf ", &x[i - 1]);
#else
      scanf("%lf ", &x[i - 1]);
#endif
    for (i = 1; i <= nxy; ++i)
#ifdef _WIN32
      scanf_s("%lf ", &y[i - 1]);
#else
      scanf("%lf ", &y[i - 1]);
#endif

    /* Set parameters for call to nag_tsa_spectrum_bivar (g13cdc) */
    /* Mean correction and 10 percent taper */
    pxy = 0.1;
    /* Window shape parameter and zero covariance at lag 16 */
    pw = 0.5;
    mw = 16;
    /* Alignment shift of 3 */
    is = 3;

    /* nag_tsa_spectrum_bivar (g13cdc).
     * Multivariate time series, smoothed sample cross spectrum
     * using spectral smoothing by the trapezium frequency
     * (Daniell) window
     */
    nag_tsa_spectrum_bivar(nxy, Nag_Mean, pxy, mw, is, pw, l, kc, x, y, &g,
                           &ng, &fail);
    if (fail.code != NE_NOERROR) {
      printf("Error from nag_tsa_spectrum_bivar (g13cdc).\n%s\n",
             fail.message);
```

```
      exit_status = 1;
      goto END;
    }

    printf("\n                       Returned sample spectrum\n");
    printf("\n       Real  Imaginary       Real  Imaginary   "
           "    Real  Imaginary\n");
    printf("      part     part         part     part     "
           "    part     part\n\n");
    for (j = 1; j <= ng; ++j)
      printf("%5" NAG_IFMT "%8.4f%9.4f%s",
             /* nag_complex_real (a02bbc).
              * Real part of a complex number
              */
             j, nag_complex_real(g[j - 1]), a02bcc(g[j - 1]),
             (j % 3 == 0 ? "\n" : ""));
    printf("\n");
    NAG_FREE(g);
  }
END:
  NAG_FREE(x);
  NAG_FREE(y);
  return exit_status;
}
```

10.2 Program Data

```
nag_tsa_spectrum_bivar (g13cdc) Example Program Data
296
-0.109  0.000  0.178  0.339  0.373  0.441  0.461  0.348
 0.127 -0.180 -0.588 -1.055 -1.421 -1.520 -1.302 -0.814
-0.475 -0.193  0.088  0.435  0.771  0.866  0.875  0.891
 0.987  1.263  1.775  1.976  1.934  1.866  1.832  1.767
 1.608  1.265  0.790  0.360  0.115  0.088  0.331  0.645
 0.960  1.409  2.670  2.834  2.812  2.483  1.929  1.485
 1.214  1.239  1.608  1.905  2.023  1.815  0.535  0.122
 0.009  0.164  0.671  1.019  1.146  1.155  1.112  1.121
 1.223  1.257  1.157  0.913  0.620  0.255 -0.280 -1.080
-1.551 -1.799 -1.825 -1.456 -0.944 -0.570 -0.431 -0.577
-0.960 -1.616 -1.875 -1.891 -1.746 -1.474 -1.201 -0.927
-0.524  0.040  0.788  0.943  0.930  1.006  1.137  1.198
 1.054  0.595 -0.080 -0.314 -0.288 -0.153 -0.109 -0.187
-0.255 -0.299 -0.007  0.254  0.330  0.102 -0.423 -1.139
-2.275 -2.594 -2.716 -2.510 -1.790 -1.346 -1.081 -0.910
-0.876 -0.885 -0.800 -0.544 -0.416 -0.271  0.000  0.403
 0.841  1.285  1.607  1.746  1.683  1.485  0.993  0.648
 0.577  0.577  0.632  0.747  0.999  0.993  0.968  0.790
 0.399 -0.161 -0.553 -0.603 -0.424 -0.194 -0.049  0.060
 0.161  0.301  0.517  0.566  0.560  0.573  0.592  0.671
 0.933  1.337  1.460  1.353  0.772  0.218 -0.237 -0.714
-1.099 -1.269 -1.175 -0.676  0.033  0.556  0.643  0.484
 0.109 -0.310 -0.697 -1.047 -1.218 -1.183 -0.873 -0.336
 0.063  0.084  0.000  0.001  0.209  0.556  0.782  0.858
 0.918  0.862  0.416 -0.336 -0.959 -1.813 -2.378 -2.499
-2.473 -2.330 -2.053 -1.739 -1.261 -0.569 -0.137 -0.024
-0.050 -0.135 -0.276 -0.534 -0.871 -1.243 -1.439 -1.422
-1.175 -0.813 -0.634 -0.582 -0.625 -0.713 -0.848 -1.039
-1.346 -1.628 -1.619 -1.149 -0.488 -0.160 -0.007 -0.092
-0.620 -1.086 -1.525 -1.858 -2.029 -2.024 -1.961 -1.952
-1.794 -1.302 -1.030 -0.918 -0.798 -0.867 -1.047 -1.123
-0.876 -0.395  0.185  0.662  0.709  0.605  0.501  0.603
 0.943  1.223  1.249  0.824  0.102  0.025  0.382  0.922
 1.032  0.866  0.527  0.093 -0.458 -0.748 -0.947 -1.029
-0.928 -0.645 -0.424 -0.276 -0.158 -0.033  0.102  0.251
 0.280  0.000 -0.493 -0.759 -0.824 -0.740 -0.528 -0.204
 0.034  0.204  0.253  0.195  0.131  0.017 -0.182 -0.262
53.8 53.6 53.5 53.5 53.4 53.1 52.7 52.4 52.2 52.0 52.0 52.4 53.0 54.0 54.9 56.0
56.8 56.8 56.4 55.7 55.0 54.3 53.2 52.3 51.6 51.2 50.8 50.5 50.0 49.2 48.4 47.9
47.6 47.5 47.5 47.6 48.1 49.0 50.0 51.1 51.8 51.9 51.7 51.2 50.0 48.3 47.0 45.8
45.6 46.0 46.9 47.8 48.2 48.3 47.9 47.2 47.2 48.1 49.4 50.6 51.5 51.6 51.2 50.5
```

```
50.1 49.8 49.6 49.4 49.3 49.2 49.3 49.7 50.3 51.3 52.8 54.4 56.0 56.9 57.5 57.3
56.6 56.0 55.4 55.4 56.4 57.2 58.0 58.4 58.4 58.1 57.7 57.0 56.0 54.7 53.2 52.1
51.6 51.0 50.5 50.4 51.0 51.8 52.4 53.0 53.4 53.6 53.7 53.8 53.8 53.8 53.3 53.0
52.9 53.4 54.6 56.4 58.0 59.4 60.2 60.0 59.4 58.4 57.6 56.9 56.4 56.0 55.7 55.3
55.0 54.4 53.7 52.8 51.6 50.6 49.4 48.8 48.5 48.7 49.2 49.8 50.4 50.7 50.9 50.7
50.5 50.4 50.2 50.4 51.2 52.3 53.2 53.9 54.1 54.0 53.6 53.2 53.0 52.8 52.3 51.9
51.6 51.6 51.4 51.2 50.7 50.0 49.4 49.3 49.7 50.6 51.8 53.0 54.0 55.3 55.9 55.9
54.6 53.5 52.4 52.1 52.3 53.0 53.8 54.6 55.4 55.9 55.9 55.2 54.4 53.7 53.6 53.6
53.2 52.5 52.0 51.4 51.0 50.9 52.4 53.5 55.6 58.0 59.5 60.0 60.4 60.5 60.2 59.7
59.0 57.6 56.4 55.2 54.5 54.1 54.1 54.4 55.5 56.2 57.0 57.3 57.4 57.0 56.4 55.9
55.5 55.3 55.2 55.4 56.0 56.5 57.1 57.3 56.8 55.6 55.0 54.1 54.3 55.3 56.4 57.2
57.8 58.3 58.6 58.8 58.8 58.6 58.0 57.4 57.0 56.4 56.3 56.4 56.4 56.0 55.2 54.0
53.0 52.0 51.6 51.6 51.1 50.4 50.0 50.0 52.0 54.0 55.1 54.5 52.8 51.4 50.8 51.2
52.0 52.8 53.8 54.5 54.9 54.9 54.8 54.4 53.7 53.3 52.8 52.6 52.6 53.0 54.3 56.0
57.0 58.0 58.6 58.5 58.3 57.8 57.3 57.0
```

10.3 Program Results

```
nag_tsa_spectrum_bivar (g13cdc) Example Program Results

                          Returned sample spectrum

        Real  Imaginary        Real  Imaginary        Real  Imaginary
        part      part         part      part         part      part

     1 -6.1563    0.0000    2 -5.5905   -2.0119    3 -3.2711   -2.7963
     4 -1.1803   -2.3264    5 -0.2061   -1.8132    6  0.3434   -1.1357
     7  0.6200   -0.7351    8  0.5967   -0.3449    9  0.4523   -0.0984
    10  0.2391    0.0177   11  0.1129    0.0402   12  0.0564    0.0523
    13  0.0134    0.0443   14 -0.0032    0.0299   15 -0.0057    0.0148
    16 -0.0057    0.0069   17 -0.0033    0.0038   18 -0.0011    0.0012
    19 -0.0004    0.0001   20 -0.0004    0.0002   21 -0.0003    0.0001
    22 -0.0001    0.0002   23 -0.0002    0.0003   24 -0.0002    0.0002
    25 -0.0002    0.0000   26 -0.0004    0.0000   27 -0.0002   -0.0002
    28 -0.0001   -0.0000   29 -0.0001    0.0002   30 -0.0001    0.0002
    31 -0.0002    0.0003   32 -0.0002    0.0001   33 -0.0001    0.0000
    34 -0.0000   -0.0000   35  0.0000   -0.0001   36  0.0001   -0.0001
    37  0.0001   -0.0001   38  0.0001   -0.0001   39  0.0000   -0.0001
    40  0.0000   -0.0001   41  0.0001    0.0000
```