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

```c
#include <nag.h>
#include <naggl3.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

\[
\hat{f}_{xy}(\omega) = \frac{1}{2\pi n} \left\{ \sum_{t=1}^{n} y_t \exp(i\omega t) \right\} \times \left\{ \sum_{t=1}^{n} x_t \exp(-i\omega t) \right\}
\]

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 \(c_f(\nu_l)\), and imaginary part or quadrature spectrum \(q_f(\nu_l)\) are defined by

\[
f_{xy}(\nu_l) = c_f(\nu_l) + iq_f(\nu_l) = \sum_{|\omega|<\pi/M} \hat{w}_k \hat{f}_{xy}(\nu_l + \omega_k)
\]

where the weights \(\hat{w}_k\) 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:

\[
\hat{w}_k = w_k \exp(-2\pi iSk/L).
\]

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 \(\frac{\pi}{M}\), 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


5 Arguments

1: \texttt{nxy} – Integer \hspace{1cm} \textit{Input}
   
   \textit{On entry:} the length of the time series \(x\) and \(y\), \(n\).
   
   \textit{Constraint:} \(n \geq 1\).

2: \texttt{mt\_correction} – NagMeanOrTrend \hspace{1cm} \textit{Input}
   
   \textit{On entry:} whether the data are to be initially mean or trend corrected.
   \texttt{mt\_correction} = Nag\_NoCorrection for no correction, \texttt{mt\_correction} = Nag\_Mean for mean correction, \texttt{mt\_correction} = Nag\_Trend for trend correction.
   
   \textit{Constraint:} \texttt{mt\_correction} = Nag\_NoCorrection, Nag\_Mean or Nag\_Trend.

3: \texttt{pxy} – double \hspace{1cm} \textit{Input}
   
   \textit{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.
   
   \textit{Constraint:} \(0.0 \leq \texttt{pxy} \leq 1.0\).

4: \texttt{mw} – Integer \hspace{1cm} \textit{Input}
   
   \textit{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.
   
   \textit{Constraint:} \(1 \leq \texttt{mw} \leq \texttt{nxy}\).

5: \texttt{is} – Integer \hspace{1cm} \textit{Input}
   
   \textit{On entry:} the alignment shift, \(S\), between the \(x\) and \(y\) series. If \(x\) leads \(y\), the shift is positive.
   
   \textit{Constraint:} \(-1 < \texttt{is} < 1\).

6: \texttt{pw} – double \hspace{1cm} \textit{Input}
   
   \textit{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 \(\texttt{mw} = \texttt{nxy}\) (i.e., no smoothing is carried out) then \texttt{pw} is not used.
   
   \textit{Constraint:} \(0.0 \leq \texttt{pw} \leq 1.0\) if \(\texttt{mw} \neq \texttt{nxy}\).

7: \texttt{l} – Integer \hspace{1cm} \textit{Input}
   
   \textit{On entry:} the frequency division, \(L\), of smoothed cross spectral estimates as \(2\pi/L\).
   
   \textit{Constraint:} \(l \geq 1\).
   
   \(l\) must be a factor of \(\texttt{kc}\) (see below).

8: \texttt{kc} – Integer \hspace{1cm} \textit{Input}
   
   \textit{On entry:} the order of the fast Fourier transform (FFT) used to calculate the spectral estimates. \texttt{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 \hspace{1cm} Input
On entry: the \( nxy \) data points of the \( x \) series.

10: \[ y|kC| \] – const double \hspace{1cm} Input
On entry: the \( nxy \) data points of the \( y \) series.

11: \[ g \] – Complex ** \hspace{1cm} 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 * \hspace{1cm} Output
On exit: the number of spectral estimates, \( [L/2] + 1 \), whose separate parts are held in \( g \).

13: \[ fail \] – NagError * \hspace{1cm} Input/Output
The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_2_INT_ARG_CONS
On entry, \( kC = \langle value \rangle \) while \( l = \langle value \rangle \). These arguments must satisfy \( kC \% l \neq 0 \) when \( l > 0 \).
On entry, \( kC = \langle value \rangle \) while \( nxy = \langle value \rangle \). These arguments must satisfy \( kC \geq 2^nxy \) when \( nxy > 0 \).
On entry, \( l = \langle value \rangle \) while \( is = \langle value \rangle \). These arguments must satisfy \( ||is|| < 1 \) when \( l > 0 \).

NE_2_INT_ARG_GT
On entry, \( mw = \langle value \rangle \) while \( nxy = \langle value \rangle \). These arguments must satisfy \( mw \leq nxy \).

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, \( l = \langle value \rangle \).
Constraint: \( l \geq 1 \).
On entry, \( mw = \langle \text{value} \rangle \).
Constraint: \( mw \geq 1 \).
On entry, \( nxy = \langle \text{value} \rangle \).
Constraint: \( nxy \geq 1 \).

**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: \( pw = \langle \text{value} \rangle \).
On entry, \( pxy \) must not be greater than 1.0: \( pxy = \langle \text{value} \rangle \).

**NE_REAL_ARG_LT**
On entry, \( pw \) must not be less than 0.0: \( pw = \langle \text{value} \rangle \).
On entry, \( pxy \) must not be less than 0.0: \( pxy = \langle \text{value} \rangle \).

**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**
Not applicable.

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 \( kc \times \log(kc) \) (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. */
* Copyright 2014 Numerical Algorithms Group.
* Mark 4, 1996.
*/
#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
```c
#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("%"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
            if (i <= nxy; ++i)
            {
                #ifdef _WIN32
                    scanf_s("%lf ", &y[i - 1]);
                #else
                    scanf("%lf ", &y[i - 1]);
                #endif
                for (i = 1; i <= nxy; ++i)
                {
                    #ifdef _WIN32
                        scanf_s("%lf ", &x[i - 1]);
                    #else
                        scanf("%lf ", &x[i - 1]);
                    #endif
                    if (fail.code != NE_NOERROR)
                    {
                        /* 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)
                        {
```
### 10.2 Program Data

**nag_tsa_spectrum_bivar (g13cdc) Example Program Data**

| Sample 1 | Sample 2 | Sample 3 | Sample 4 | Sample 5 | Sample 6 | Sample 7 | Sample 8 | Sample 9 | Sample 10 | Sample 11 | Sample 12 | Sample 13 | Sample 14 | Sample 15 | Sample 16 |
|----------|----------|----------|----------|----------|----------|----------|----------|----------|-----------|-----------|-----------|-----------|-----------|-----------|----------|----------|
| 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    |
| 1.755    | 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   | -1.554   | 0.716    | -2.510   | -1.790   | -1.346   | -1.081   | -0.910   | -0.876   | -0.885   | -0.800   | -0.544   | -0.416   | -0.271   | 0.000    | 0.401    |
| 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    |
| -1.099   | -1.310   | -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     |
| 56.8     | 56.8     | 56.4     | 55.7     | 55.0     | 54.3     | 53.2     | 52.3     | 51.9     | 51.6     | 51.5     | 51.5     | 51.2     | 50.8     | 50.5     | 50.0     | 49.2     | 48.4     | 47.9     |

**Note:** The program data is printed in a formatted manner to fit the text format. The actual data values are listed in the table above.
### Program Results

**nag_tsa_spectrum_bivar (g13cdc) Example Program Results**

<table>
<thead>
<tr>
<th>Returned sample spectrum</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Real part</strong></td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>13</td>
</tr>
<tr>
<td>16</td>
</tr>
<tr>
<td>19</td>
</tr>
<tr>
<td>22</td>
</tr>
<tr>
<td>25</td>
</tr>
<tr>
<td>28</td>
</tr>
<tr>
<td>31</td>
</tr>
<tr>
<td>34</td>
</tr>
<tr>
<td>37</td>
</tr>
<tr>
<td>40</td>
</tr>
</tbody>
</table>