NAG Library Function Document
nag_tsa_spectrum_univar (g13cbc)

1 Purpose
nag_tsa_spectrum_univar (g13cbc) calculates the smoothed sample spectrum of a univariate time series using spectral smoothing by the trapezium frequency (Daniell) window.

2 Specification
#include <nag.h>
#include <nagg13.h>

void nag_tsa_spectrum_univar (Integer nx, NagMeanOrTrend mt_correction,
    double px, Integer mw, double pw, Integer l, Integer kc,
    Nag_LoggedSpectra lg_spect, const double x[], double **g, Integer *ng,
    double stats[], NagError *fail)

3 Description
The supplied time series may be mean or trend corrected (by least squares), and tapered, the tapering factors being those of the split cosine bell:
\[
\frac{1}{2} \left( 1 - \cos \left( \frac{n(t-\frac{1}{2})}{T} \right) \right), \quad \text{for } 1 \leq t \leq T
\]
\[
\frac{1}{2} \left( 1 - \cos \left( \frac{n(n+\frac{1}{2})}{T} \right) \right), \quad \text{for } n+1-T \leq t \leq n
\]
otherwise
where \( T = \frac{np}{2} \) and \( p \) is the tapering proportion.

The unsmoothed sample spectrum
\[
f^*(\omega) = \frac{1}{2} \pi \left| \sum_{t=1}^{n} x_t \exp(i\omega t) \right|^2
\]
is then calculated for frequency values
\[
\omega_k = \frac{2\pi k}{K}, k = 0, 1, \ldots, \lfloor K/2 \rfloor
\]
where \([ \ ]\) denotes the integer part.

The smoothed spectrum is returned as a subset of these frequencies for which \( K \) is a multiple of a chosen value \( r \), i.e.,
\[
\omega_{rl} = \nu_l = \frac{2\pi l}{L}, l = 0, 1, \ldots, \lfloor L/2 \rfloor
\]
where \( K = r \times L \). You will normally fix \( L \) first, then choose \( r \) so that \( K \) is sufficiently large to provide an adequate representation for the unsmoothed spectrum, i.e., \( K \geq 2 \times n \). It is possible to take \( L = K \), i.e., \( r = 1 \).

The smoothing is defined by a trapezium window whose shape is supplied by the function
\[
W(\alpha) = \begin{cases} 1, & |\alpha| \leq p \\ \frac{|\alpha|}{1-p}, & p < |\alpha| \leq 1 \end{cases}
\]
the proportion \( p \) being supplied by you.
The width of the window is fixed as \(2\pi/M\) by supplying \(M\). A set of averaging weights are constructed:

\[W_k = g \times W\left(\frac{\omega_k M}{\pi}\right), 0 \leq \omega_k \leq \frac{\pi}{M}\]

where \(g\) is a normalizing constant, and the smoothed spectrum obtained is

\[\hat{f}(\nu) = \sum_{|\omega| < \pi} W_k f^*(\nu + \omega_k)\]

If no smoothing is required \(M\) should be set to \(n\), in which case the values returned are \(\hat{f}(\nu) = f^*(\nu)\). Otherwise, in order that the smoothing approximates well to an integration, it is essential that \(K \gg M\), and preferable, but not essential, that \(K\) be a multiple of \(M\). A choice of \(L > M\) would normally be required to supply an adequate description of the smoothed spectrum. Typical choices of \(L \simeq n\) and \(K \simeq 4n\) should be adequate for usual smoothing situations when \(M < n/5\).

The sampling distribution of \(\hat{f}(\omega)\) is approximately that of a scaled \(\chi^2_d\) variate, whose degrees of freedom \(d\) is provided by the function, together with multiplying limits \(mu, ml\) from which approximate 95% confidence intervals for the true spectrum \(f(\omega)\) may be constructed as \([ml \times \hat{f}(\omega), mu, \times, \hat{f}(\omega)]\).

Alternatively, \(\log \hat{f}(\omega)\) may be returned, with additive limits.

The bandwidth \(b\) of the corresponding smoothing window in the frequency domain is also provided. Spectrum estimates separated by (angular) frequencies much greater than \(b\) may be assumed to be independent.

4 References


5 Arguments

1: \(\text{nx} \rightarrow \text{Integer} \quad \text{Input}\)

   *On entry:* the length of the time series, \(n\).

   *Constraint:* \(\text{nx} \geq 1\).

2: \(\text{mt_correction} \rightarrow \text{NagMeanOrTrend} \quad \text{Input}\)

   *On entry:* whether the data are to be initially mean or trend corrected.

   \(\text{mt_correction} = \text{Nag_NoCorrection}\) for no correction, \(\text{mt_correction} = \text{Nag_Mean}\) for mean correction, \(\text{mt_correction} = \text{Nag_Trend}\) for trend correction.

   *Constraint:* \(\text{mt_correction} = \text{Nag_NoCorrection}, \text{Nag_Mean}\) or \(\text{Nag_Trend}\).

3: \(\text{px} \rightarrow \text{double} \quad \text{Input}\)

   *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:* \(0.0 \leq \text{px} \leq 1.0\).

4: \(\text{mw} \rightarrow \text{Integer} \quad \text{Input}\)

   *On entry:* the value of \(M\) which determines the frequency width of the smoothing window as \(2\pi/M\). A value of \(n\) implies no smoothing is to be carried out.

   *Constraint:* \(1 \leq \text{mw} \leq \text{nx}\).

5: \(\text{pw} \rightarrow \text{double} \quad \text{Input}\)

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

\[ \text{Constraint: } 0.0 \leq pw \leq 1.0 \text{ if } mw \neq nx. \]

6: \( l \) – Integer \( Input \)

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

Constraints:
\[ l \geq 1; \]
\[ l \text{ must be a factor of } kc \text{ (see below)}. \]

7: \( kc \) – Integer \( Input \)

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

Constraints:
\[ kc \geq 2 \times nx; \]
\( kc \) 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.

8: \( lg\text{spect} \) – Nag_LoggedSpectra \( Input \)

On entry: indicates whether unlogged or logged spectral estimates and confidence limits are required. \( lg\text{spect} = \text{Nag_Unlogged} \) for unlogged. \( lg\text{spect} = \text{NagLogged} \) for logged.

Constraint: \( lg\text{spect} = \text{NagUnlogged} \) or \( \text{NagLogged} \).

9: \( x[kc] \) – const double \( Input \)

On entry: the \( n \) data points.

10: \( g \) – double ** \( Output \)

On exit: vector which contains the \( ng \) spectral estimates \( \hat{f}(\omega_i) \), for \( i = 0, 1, \ldots, [L/2] \), in \( g[0] \) to \( g[ng - 1] \) (logged if \( lg\text{spect} = \text{NagLogged} \)). 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 \( \text{NAG_FREE} \) should be used to free the memory in between calls.

11: \( ng \) – Integer * \( Output \)

On exit: the number of spectral estimates, \( [L/2] + 1 \), in \( g \).

12: \( \text{stats}[4] \) – double \( Output \)

On exit: four associated statistics. These are the degrees of freedom in \( \text{stats}[0] \), the lower and upper 95% confidence limit factors in \( \text{stats}[1] \) and \( \text{stats}[2] \) respectively (logged if \( lg\text{spect} = \text{NagLogged} \)), and the bandwidth in \( \text{stats}[3] \).

13: \( \text{fail} \) – NagError * \( Input/Output \)

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 \(kc\%l = 0\) when \(l > 0\).
On entry, \(kc = \langle value\rangle\) while \(nx = \langle value\rangle\). These arguments must satisfy \(kc \geq 2 \times nx\) when \(nx > 0\).

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

NE_ALLOC_FAIL
Dynamic memory allocation failed.

NE_BAD_PARAM
On entry, argument \(lg\_spect\) had an illegal value.
On entry, argument \(mt\_correction\) had an illegal value.

NE_CONFID_LIMIT_FACT
The calculation of confidence limit factors has failed. Spectral estimates (logged if requested) are returned in \(g\), and degrees of freedom and bandwidth in \(stats\).

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, \(mw = \langle value\rangle\).
Constraint: \(mw \geq 1\).
On entry, \(nx = \langle value\rangle\).
Constraint: \(nx \geq 1\).
On entry, \(pw\) must not be less than 0.0: \(pw = \langle value\rangle\).

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, \(px\) must not be greater than 1.0: \(px = \langle value\rangle\).

NE_REAL_ARG_LT
On entry, \(px\) must not be less than 0.0: \(px = \langle value\rangle\).

NE_SPECTRAL_ESTIM_NEG
One or more spectral estimates are negative. Unlogged spectral estimates are returned in \(g\) and the degrees of freedom, unlogged confidence limit factors and bandwidth in \(stats\).
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_univar (g13cbc) carries out a FFT of length \( kc \) to calculate the sample spectrum. The time taken by the function for this is approximately proportional to \( kc \times \log(kc) \) (but see Section 9 in nag_sum_fft_realherm_1d (c06pac) for further details).

10 Example

The example program reads a time series of length 131. It selects the mean correction option, a tapering proportion of 0.2, the option of no smoothing and a frequency division for logged spectral estimates of \( 2\pi/100 \). It then calls nag_tsa_spectrum_univar (g13cbc) to calculate the univariate spectrum and prints the logged spectrum together with 95% confidence limits. The program then selects a smoothing window with frequency width \( 2\pi/30 \) and shape argument 0.5 and recalculates and prints the logged spectrum and 95% confidence limits.

10.1 Program Text

```c
/* nag_tsa_spectrum_univar (g13cbc) Example Program. */
* Copyright 2014 Numerical Algorithms Group.
* Mark 4, 1996.
*/
#include <nag.h>
#include <nag_stdlib.h>
#include <stdio.h>
#include <naggl3.h>
#define KCMAX 400
#define NMAX KCMAX/2
int main(void)
{
  Integer exit_status = 0, i, kc, l, mw, ng, nx;
  NagError fail;
  double *g, pw, px, *stats = 0, *x = 0, *xh = 0;
  INIT_FAIL(fail);
  printf("nag_tsa_spectrum_univar (g13cbc) Example Program Results\n");
  /* Skip heading in data file */
  #ifdef _WIN32
    scanf_s("%*[\n"]);
  #else
    scanf("%*[\n"]);
  #endif
```
ifdef _WIN32
    scanf_s("%"NAG_IFMT" ", &nx);
#else
    scanf("%"NAG_IFMT" ", &nx);
#endif
if (nx >= 1 && nx <= NXMAX)
{
    if (!(stats = NAG_ALLOC(4, double)) ||
        !(x = NAG_ALLOC(KCMAX, double)) ||
        !(xh = NAG_ALLOC(NXMAX, double))
    )
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }
}
else
{
    printf("Invalid nx.\n");
    exit_status = 1;
    return exit_status;
}
for (i = 1; i <= nx; ++i)
#ifdef _WIN32
    scanf_s("%lf ", &xh[i - 1]);
#else
    scanf("%lf ", &xh[i - 1]);
#endif
px = 0.2;
mw = nx;
pw = 0.5;
kc = KCMAX;
l = 100;
#ifdef _WIN32
    while ((scanf_s("%"NAG_IFMT" ", &mw)) != EOF)
#else
    while ((scanf("%"NAG_IFMT" ", &mw)) != EOF)
#endif
{
    if (mw > 0 && mw <= nx)
    {
        for (i = 1; i <= nx; ++i)
            x[i - 1] = xh[i - 1];
    /* nag_tsa_spectrum_univar (g13cbc).
    * Univariate time series, smoothed sample spectrum using
    * spectral smoothing by the trapezium frequency (Daniell)
    * window
    */
        nag_tsa_spectrum_univar(nx, Nag_Mean, px, mw, pw, l, kc, Nag_Logged,
            x, &g, &ng, stats, &fail);
        if (fail.code != NE_NOERROR)
        {
            printf(
"Error from nag_tsa_spectrum_univar (g13cbc).
 fail.message);"
exit_status = 1;
        goto END;
    }
    if (mw == nx)
        printf("\n No smoothing\n");
    else
        printf("\n Frequency width of smoothing window = \
"1/%"NAG_IFMT"\n", mw);
    printf(
" Degrees of freedom =%4.1f Bandwidth =%7.4f\n",
                stats[0], stats[3]);
    printf("95 percent confidence limits - Lower =%7.4f "
"Upper =%7.4f\n", stats[1], stats[2]);
printf(" Spectrum Spectrum Spectrum"
);
printf(" estimate estimate estimate"

);
for (i = 1; i <= ng; ++i)
printf("%5"NAG_IFMT"%10.4f%s", i, g[i - 1],
     (i%4 == 0?"\n":""));
printf("\n");
NAG_FREE(g);

}

END:
NAG_FREE(stats);
NAG_FREE(x);
NAG_FREE(xh);
return exit_status;

10.2 Program Data

nag_tsa_spectrum_univar (g13cbc) Example Program Data

131
  8.080  8.244  8.490  8.867  9.469  9.786 10.100 10.714
 10.787 11.000 11.133 11.100 11.800 12.250 11.350 11.575
 10.767  9.825  9.150
131
30

10.3 Program Results

nag_tsa_spectrum_univar (g13cbc) Example Program Results

No smoothing

Degrees of freedom = 2.0    Bandwidth = 0.0480

95 percent confidence limits -   Lower =-1.3053   Upper = 3.6762

| Frequency width of smoothing window | 1/30 |

Mark 25
Degrees of freedom = 7.0  Bandwidth = 0.1767

95 percent confidence limits -  Lower = -0.8275  Upper = 1.4213

<table>
<thead>
<tr>
<th>Spectrum estimate</th>
<th>Spectrum estimate</th>
<th>Spectrum estimate</th>
<th>Spectrum estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 -0.1776</td>
<td>2 -0.4561</td>
<td>3 -0.1784</td>
<td>4 1.9042</td>
</tr>
<tr>
<td>5 2.1094</td>
<td>6 1.7061</td>
<td>7 -0.7659</td>
<td>8 -1.4734</td>
</tr>
<tr>
<td>9 -1.5939</td>
<td>10 -2.1157</td>
<td>11 -2.9151</td>
<td>12 -2.7055</td>
</tr>
<tr>
<td>17 -4.0601</td>
<td>18 -4.4756</td>
<td>19 -4.2700</td>
<td>20 -4.3092</td>
</tr>
<tr>
<td>21 -4.5711</td>
<td>22 -4.8111</td>
<td>23 -4.5658</td>
<td>24 -4.7285</td>
</tr>
<tr>
<td>25 -5.4386</td>
<td>26 -5.5081</td>
<td>27 -5.2325</td>
<td>28 -5.0262</td>
</tr>
<tr>
<td>29 -4.4539</td>
<td>30 -4.4764</td>
<td>31 -4.9152</td>
<td>32 -5.8492</td>
</tr>
<tr>
<td>33 -5.5872</td>
<td>34 -4.9804</td>
<td>35 -4.8904</td>
<td>36 -5.2666</td>
</tr>
<tr>
<td>37 -5.7643</td>
<td>38 -5.8620</td>
<td>39 -5.5011</td>
<td>40 -5.7129</td>
</tr>
<tr>
<td>41 -6.3894</td>
<td>42 -6.4027</td>
<td>43 -6.1352</td>
<td>44 -6.5766</td>
</tr>
<tr>
<td>45 -7.3676</td>
<td>46 -7.1405</td>
<td>47 -6.1674</td>
<td>48 -5.8600</td>
</tr>
<tr>
<td>49 -6.1036</td>
<td>50 -6.2673</td>
<td>51 -6.4321</td>
<td></td>
</tr>
</tbody>
</table>