# NAG Toolbox for MATLAB

# g13cd

## 1 Purpose

g13cd calculates the smoothed sample cross spectrum of a bivariate time series using spectral smoothing by the trapezium frequency (Daniell) window.

## 2 Syntax

```
[xg, yg, ng, ifail] = g13cd(nxy, mtxy, pxy, mw, ish, pw, l, xg, yg,
'kc', kc)
```

## 3 Description

The supplied time series may be mean and trend corrected and tapered as in the description of g13cb 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 g13cb for univariate frequency window smoothing, the smoothed spectrum is returned as a subset of these frequencies,

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

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

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

where the weights $\tilde{w}_k$ are similar to the weights $w_k$ defined for g13cb, but allow for an implicit alignment shift $S$ between the series:

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

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 $\frac{2\pi}{M}$, should be set to $n$.

The bandwidth of the estimates will normally have been calculated in a previous call of g13cb 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 Parameters

## 5.1 Compulsory Input Parameters

1: **nxy – int32 scalar**

$n$, the length of the time series $x$ and $y$.

*Constraint*: $\mathbf{nxy} \geq 1$.

2: **mtxy – int32 scalar**

Whether the data is to be initially mean or trend corrected.

$\mathbf{mtxy} = 0$

For no correction.

$\mathbf{mtxy} = 1$

For mean correction.

$\mathbf{mtxy} = 2$

For trend correction.

*Constraint*: $0 \leq \mathbf{mtxy} \leq 2$.

3: **pxy – double scalar**

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*: $0.0 \leq \mathbf{pxy} \leq 1.0$.

4: **mw – int32 scalar**

$M$, the frequency width of the smoothing window as $\frac{2\pi}{M}$.

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

*Constraint*: $1 \leq \mathbf{mw} \leq \mathbf{nxy}$.

5: **ish – int32 scalar**

$S$, the alignment shift between the $x$ and $y$ series. If $x$ leads $y$, the shift is positive.

*Constraint*: $-\mathbf{l} < \mathbf{ish} < \mathbf{l}$.

6: **pw – double scalar**

$p$, the shape parameter of the trapezium frequency window.

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

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

*Constraint*: if $\mathbf{mw} \neq \mathbf{nxy}$, $0.0 \leq \mathbf{pw} \leq 1.0$.

7: **l – int32 scalar**

$L$, the frequency division of smoothed cross spectral estimates as $\frac{2\pi}{L}$.

*Constraints*:

$\mathbf{l} \geq 1$;
**l** must be a factor of **kc**.

8: **xg(kc) – double array**

The **nxy** data points of the $x$ series.

9: **yg(kc) – double array**

The **nxy** data points of the $y$ series.

### 5.2 Optional Input Parameters

1: **kc – int32 scalar**

*Default*: The dimension of the arrays **xg**, **yg**. (An error is raised if these dimensions are not equal.)

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 c06ea and c06eb which perform the FFT.

### 5.3 Input Parameters Omitted from the MATLAB Interface

None.

### 5.4 Output Parameters

1: **xg(kc) – double array**

The real parts of the **ng** cross spectral estimates in elements **xg**(1) to **xg**(**ng**), and **xg**(**ng** + 1) to **xg**(**kc**) contain 0.0. The $y$ series leads the $x$ series.

2: **yg(kc) – double array**

The imaginary parts of the **ng** cross spectral estimates in elements **yg**(1) to **yg**(**ng**), and **yg**(**ng** + 1) to **yg**(**kc**) contain 0.0. The $y$ series leads the $x$ series.

3: **ng – int32 scalar**

The number of spectral estimates, $[L/2] + 1$, whose separate parts are held in **xg** and **yg**.

4: **ifail – int32 scalar**

0 unless the function detects an error (see Section 6).

## 6 Error Indicators and Warnings

Errors or warnings detected by the function:

**ifail** = 1

On entry, **nxy** $< 1$,
or **mtxy** $< 0$,
or **mtxy** $> 2$,
or **pxy** $< 0.0$,
or **pxy** $> 1.0$,
or **mw** $< 1$,
or **mw** $>$ **nxy**,
or **pw** $< 0.0$ and **mw** $\neq$ **nxy**,
or **pw** $> 1.0$ and **mw** $\neq$ **nxy**,

or $\mathbf{l} < 1$,
or $|\mathbf{ish}| \geq \mathbf{l}$.

**ifail** $= 2$

On entry, $\mathbf{kc} < 2 \times \mathbf{nxy}$,
or **kc** is not a multiple of **l**,
or **kc** has a prime factor exceeding 19,
or **kc** has more than 20 prime factors, counting repetitions.

**ifail** $= 3$

This indicates that a serious error has occurred. Check all array subscripts in calls to g13cd. Seek expert help.

## 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 Further Comments

g13cd 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 $\mathbf{kc} \times \log(\mathbf{kc})$ (but see function document c06ea for further details).

## 9 Example

```
nxy = int32(296);
mtxy = int32(1);
pxy = 0.1;
mw = int32(16);
ish = int32(3);
pw = 0.5;
l = int32(80);
xg = zeros(640, 1);
xg(1:296) = [-0.109;
     0;
     0.178;
     0.339;
     0.373;
     0.441;
     0.461;
     0.348;
     0.127;
     -0.18;
     -0.588;
     -1.055;
     -1.421;
     -1.52;
     -1.302;
     -0.8139999999999999;
     -0.475;
     -0.193;
     0.08799999999999999;
     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;
    0.79;
    0.36;
    0.115;
    0.08799999999999999;
    0.331;
    0.645;
    0.96;
    1.409;
    2.67;
    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;
    0.008999999999999999;
    0.164;
    0.671;
    1.019;
    1.146;
    1.155;
    1.112;
    1.121;
    1.223;
    1.257;
    1.157;
    0.913;
    0.62;
    0.255;
    -0.28;
    -1.08;
    -1.551;
    -1.799;
    -1.825;
    -1.456;
    -0.944;
    -0.57;
    -0.431;
    -0.577;
    -0.96;
    -1.616;
    -1.875;
    -1.891;
    -1.746;
    -1.474;
    -1.201;
    -0.927;
    -0.524;
    0.04;
    0.788;
    0.9429999999999999;
    0.93;
    1.006;
    1.137;
    1.198;
    1.054;
```

```
      0.595;
      -0.08;
      -0.314;
      -0.288;
      -0.153;
      -0.109;
      -0.187;
      -0.255;
      -0.299;
      -0.007;
      0.254;
      0.33;
      0.102;
      -0.423;
      -1.139;
      -2.275;
      -2.594;
      -2.716;
      -2.51;
      -1.79;
      -1.346;
      -1.081;
      -0.91;
      -0.876;
      -0.885;
      -0.8;
      -0.544;
      -0.416;
      -0.271;
      0;
      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;
      0.79;
      0.399;
      -0.161;
      -0.553;
      -0.603;
      -0.424;
      -0.194;
      -0.049;
      0.06;
      0.161;
      0.301;
      0.517;
      0.5659999999999999;
      0.5600000000000001;
      0.573;
      0.592;
      0.671;
      0.9330000000000001;
      1.337;
      1.46;
      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;
    -0.31;
    -0.697;
    -1.047;
    -1.218;
    -1.183;
    -0.873;
    -0.336;
    0.063;
    0.08400000000000001;
    0;
    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;
    -2.33;
    -2.053;
    -1.739;
    -1.261;
    -0.569;
    -0.137;
    -0.024;
    -0.05;
    -0.135;
    -0.276;
    -0.534;
    -0.871;
    -1.243;
    -1.439;
    -1.422;
    -1.175;
    -0.8129999999999999;
    -0.634;
    -0.582;
    -0.625;
    -0.713;
    -0.848;
    -1.039;
    -1.346;
    -1.628;
    -1.619;
    -1.149;
    -0.488;
    -0.16;
    -0.007;
    -0.092;
    -0.62;
    -1.086;
    -1.525;
    -1.858;
    -2.029;
```

```
      -2.024;
      -1.961;
      -1.952;
      -1.794;
      -1.302;
      -1.03;
      -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;
      0.9429999999999999;
      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;
      0.28;
      0;
      -0.493;
      -0.759;
      -0.824;
      -0.74;
      -0.528;
      -0.204;
      0.034;
      0.204;
      0.253;
      0.195;
      0.131;
      0.017;
      -0.182;
      -0.262];
yg = zeros(640, 1);
yg(1:296) = [53.8;
      53.6;
      53.5;
      53.5;
      53.4;
      53.1;
      52.7;
      52.4;
      52.2;
      52;
```

```
      52;
      52.4;
      53;
      54;
      54.9;
      56;
      56.8;
      56.8;
      56.4;
      55.7;
      55;
      54.3;
      53.2;
      52.3;
      51.6;
      51.2;
      50.8;
      50.5;
      50;
      49.2;
      48.4;
      47.9;
      47.6;
      47.5;
      47.5;
      47.6;
      48.1;
      49;
      50;
      51.1;
      51.8;
      51.9;
      51.7;
      51.2;
      50;
      48.3;
      47;
      45.8;
      45.6;
      46;
      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;
      56.9;
      57.5;
      57.3;
```

```
    56.6;
    56;
    55.4;
    55.4;
    56.4;
    57.2;
    58;
    58.4;
    58.4;
    58.1;
    57.7;
    57;
    56;
    54.7;
    53.2;
    52.1;
    51.6;
    51;
    50.5;
    50.4;
    51;
    51.8;
    52.4;
    53;
    53.4;
    53.6;
    53.7;
    53.8;
    53.8;
    53.8;
    53.3;
    53;
    52.9;
    53.4;
    54.6;
    56.4;
    58;
    59.4;
    60.2;
    60;
    59.4;
    58.4;
    57.6;
    56.9;
    56.4;
    56;
    55.7;
    55.3;
    55;
    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;
53.6;
53.2;
53;
52.8;
52.3;
51.9;
51.6;
51.6;
51.4;
51.2;
50.7;
50;
49.4;
49.3;
49.7;
50.6;
51.8;
53;
54;
55.3;
55.9;
55.9;
54.6;
53.5;
52.4;
52.1;
52.3;
53;
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;
51.4;
51;
50.9;
52.4;
53.5;
55.6;
58;
59.5;
60;
60.4;
60.5;
60.2;
59.7;
59;
57.6;
56.4;
55.2;
54.5;
54.1;
54.1;
54.4;
55.5;
56.2;
57;
57.3;
```

```
      57.4;
      57;
      56.4;
      55.9;
      55.5;
      55.3;
      55.2;
      55.4;
      56;
      56.5;
      57.1;
      57.3;
      56.8;
      55.6;
      55;
      54.1;
      54.3;
      55.3;
      56.4;
      57.2;
      57.8;
      58.3;
      58.6;
      58.8;
      58.8;
      58.6;
      58;
      57.4;
      57;
      56.4;
      56.3;
      56.4;
      56.4;
      56;
      55.2;
      54;
      53;
      52;
      51.6;
      51.6;
      51.1;
      50.4;
      50;
      50;
      52;
      54;
      55.1;
      54.5;
      52.8;
      51.4;
      50.8;
      51.2;
      52;
      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;
      54.3;
      56;
      57;
      58;
```

```
     58.6;
     58.5;
     58.3;
     57.8;
     57.3;
     57];
[xgOut, ygOut, ng, ifail] = g13cd(nxy, mtxy, pxy, mw, ish, pw, l, xg, yg)
```

```
xgOut =
     array elided
ygOut =
     array elided
ng =
          41
ifail =
           0
```