1 Purpose
For a bivariate time series, nag_tsa_gain_phase_bivar (g13cfc) calculates the gain and phase together with lower and upper bounds from the univariate and bivariate spectra.

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

void nag_tsa_gain_phase_bivar (const double xg[], const double yg[],
        const Complex xyg[], Integer ng, const double stats[],
        double gn[],
        double gnlw[],
        double gnup[],
        double ph[],
        double phlw[],
        double phup[],
        NagError *fail)

3 Description
Estimates of the gain $G(\omega)$ and phase $\phi(\omega)$ of the dependency of series $y$ on series $x$ at frequency $\omega$ are given by

$$\hat{G}(\omega) = \frac{\tilde{A}(\omega)}{\tilde{I}(\omega)}$$

$$\hat{\phi}(\omega) = \arccos\left(\frac{\tilde{L}(\omega)}{\tilde{I}(\omega)}\right), \quad \text{if } qf(\omega) \geq 0$$

$$\hat{\phi}(\omega) = 2\pi - \arccos\left(\frac{\tilde{L}(\omega)}{\tilde{I}(\omega)}\right), \quad \text{if } qf(\omega) < 0.$$  

The quantities used in these definitions are obtained as in Section 3 in nag_tsa_cross_spectrum_bivar (g13cec).

Confidence limits are returned for both gain and phase, but should again be taken as very approximate when the coherency $W(\omega)$, as calculated by nag_tsa_gain_phase_bivar (g13cfc), is not significant. These are based on the assumption that both $\left(\hat{G}(\omega)/G(\omega)\right) - 1$ and $\hat{\phi}(\omega)$ are Normal with variance

$$\frac{1}{d} \left(\frac{1}{W(\omega)} - 1\right).$$

Although the estimate of $\phi(\omega)$ is always given in the range $[0, 2\pi]$, no attempt is made to restrict its confidence limits to this range.

4 References

5 Arguments
1:  xg[ng] – const double Input
    On entry: the ng univariate spectral estimates, $f_{xx}(\omega)$, for the $x$ series.

2:  yg[ng] – const double Input
    On entry: the ng univariate spectral estimates, $f_{yy}(\omega)$, for the $y$ series.
3: $\text{xyg}[\text{ng}]$ – const Complex 

On entry: $f_{xy}^\omega$ the ng bivariate spectral estimates for the $x$ and $y$ series. The $x$ series leads the $y$ series.

Note: the two univariate and the bivariate spectra must each have been calculated using the same amount of smoothing. The frequency width and the shape of the window and the frequency division of the spectral estimates must be the same. The spectral estimates and statistics must also be unlogged.

4: $\text{ng}$ – Integer 

On entry: the number of spectral estimates in each of the arrays $\text{xg}$, $\text{yg}$ and $\text{xyg}$. It is also the number of gain and phase estimates.

Constraint: $\text{ng} \geq 1$.

5: $\text{stats}[4]$ – const double 

On entry: the 4 associated statistics for the univariate spectral estimates for the $x$ and $y$ series. $\text{stats}[0]$ contains the degrees of freedom, $\text{stats}[1]$ and $\text{stats}[2]$ contain the lower and upper bound multiplying factors respectively and $\text{stats}[3]$ holds the bandwidth.

Constraint: $\text{stats}[0] \geq 3.0$.

6: $\text{gn}[\text{ng}]$ – double 

On exit: the ng gain estimates, $\hat{G}(\omega)$, at each frequency $\omega$.

7: $\text{gnlw}[\text{ng}]$ – double 

On exit: the ng lower bounds for the ng gain estimates.

8: $\text{gnup}[\text{ng}]$ – double 

On exit: the ng upper bounds for the ng gain estimates.

9: $\text{ph}[\text{ng}]$ – double 

On exit: the ng phase estimates, $\hat{\phi}(\omega)$, at each frequency $\omega$.

10: $\text{phlw}[\text{ng}]$ – double 

On exit: the ng lower bounds for the ng phase estimates.

11: $\text{phup}[\text{ng}]$ – double 

On exit: the ng upper bounds for the ng phase estimates.

12: $\text{fail}$ – NagError * 

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings 

NE_ALLOC_FAIL 
Dynamic memory allocation failed.

NE_BIVAR_SPECTRAL_ESTIM_ZERO 
A bivariate spectral estimate is zero.

For this frequency the gain and the phase and their bounds are set to zero.
On entry, \(ng = \langle \text{value} \rangle\).
Constraint: \(ng \geq 1\).

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.

On entry, \(stats[0]\) must not be less than 3.0: \(stats[0] = \langle \text{value} \rangle\).

A calculated value of the squared coherency exceeds one.
For this frequency the squared coherency is reset to 1.0 in the formulae for the gain and phase bounds.

A bivariate spectral estimate is negative.
For this frequency the gain and the phase and their bounds are set to zero.

A bivariate spectral estimate is zero.
For this frequency the gain and the phase and their bounds are set to zero.

All computations are very stable and yield good accuracy.

Not applicable.

The time taken by nag_tsa_gain_phase_bivar (g13cfc) is approximately proportional to \(ng\).

The example program reads the set of univariate spectrum statistics, the 2 univariate spectra and the cross spectrum at a frequency division of \(20\) for a pair of time series. It calls nag_tsa_gain_phase_bivar (g13cfc) to calculate the gain and the phase and their bounds and prints the results.

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

/* nag_tsa_gain_phase_bivar (g13cfc) Example Program. *
 * Copyright 2014 Numerical Algorithms Group.
 * *
 * Mark 4, 1996.
 * *
 */
#include <nag.h>
#include <stdio.h>
```c
#include <nag_stdlib.h>
#include <naga02.h>
#include <nagg13.h>

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

int main(void)
{

    Complex  *xyg = 0;
    Integer  exit_status = 0, i, is, j, kc = KC, l = L, mw, ng, nxy;
    NagError fail;
    double   *gn = 0, *gnlw = 0, *gnum = 0, *ph = 0, *phlw = 0, *phup = 0, pw,
             *x = 0, *xg = 0, *y = 0, *yg = 0;

    INIT_FAIL(fail);

    printf("nag_tsa_gain_phase_bivar (g13cfc) 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 (!(stats = NAG_ALLOC(4, double)) ||
            !(x = NAG_ALLOC(KC, double)) ||
            !(y = NAG_ALLOC(KC, double)) ||
            !(gnlw = NAG_ALLOC(NGMAX, double)) ||
            !(gnum = NAG_ALLOC(NGMAX, double)) ||
            !(phlw = NAG_ALLOC(NGMAX, double)) ||
            !(phup = NAG_ALLOC(NGMAX, double)) ||
            !(gn = NAG_ALLOC(NGMAX, double)) ||
            !(ph = NAG_ALLOC(NGMAX, 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_univar (g13cbc) and g13cdc
         * with 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 */
    }

END:

```
is = 3;

/* Obtain univariate spectrum for the x and the y series */
/* nag_tsa_spectrum_univar (g13cbc).*/
/* Univariate time series, smoothed sample spectrum using */
/* spectral smoothing by the trapezium frequency (Daniell)*/
/* window */

nag_tsa_spectrum_univar(nxy, Nag_Mean, pxy, mw, pw, l, kc, Nag_Unlogged, x, &xg, &ng, stats, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_tsa_spectrum_univar (g13cbc).\n", fail.message);
    exit_status = 1;
    goto END;
}

/* nag_tsa_spectrum_univar (g13cbc), see above. */

nag_tsa_spectrum_univar(nxy, Nag_Mean, pxy, mw, pw, l, kc, Nag_Unlogged, y, &yg, &ng, stats, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_tsa_spectrum_univar (g13cbc).\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Obtain cross spectrum of the bivariate series */
/* 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, &xyg, &ng, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_tsa_spectrum_bivar (g13cdc).\n", fail.message);
    exit_status = 1;
    goto END;
}

/* nag_tsa_gain_phase_bivar (g13cfc).*/
/* Multivariate time series, gain, phase, bounds, univariate */
/* and bivariate (cross) spectra */

nag_tsa_gain_phase_bivar(xg, yg, xyg, ng, stats, gn, gnlw, gnup, ph, phlw, phup, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_tsa_gain_phase_bivar (g13cfc).\n", fail.message);
    exit_status = 1;
    goto END;
}

printf("\n");
printf(" The gain\n");
printf(" Lower Upper\n");
printf(" Value bound bound\n");
for (j = 1; j <= ng; ++j)
    printf("%6"NAG_IFMT" %10.4f %10.4f %10.4f\n", j - 1, gn[j - 1], gnlw[j - 1], gnup[j - 1]);
printf("\n The phase\n");
printf(" Lower Upper\n");
for (j = 1; j <= ng; ++j)
    printf("%6"NAG_IFMT" %10.4f %10.4f %10.4f\n", 

Mark 25


```c
j - 1, ph[j - 1], phlw[j - 1], phup[j - 1]);
NAG_FREE(xg);
NAG_FREE(yg);
NAG_FREE(xyg);
END:
NAG_FREE(stats);
NAG_FREE(x);
NAG_FREE(y);
NAG_FREE(gnlw);
NAG_FREE(gnup);
NAG_FREE(phlw);
NAG_FREE(phup);
NAG_FREE(gn);
NAG_FREE(ph);
return exit_status;
```

### 10.2 Program Data

nag_tsa_gain_phase_bivar (g13cfc) 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 -1.520 -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 0.970 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 0.364 0.717 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 0.968 0.790
0.399 -0.161 -0.553 -0.603 -0.424 -0.194 -0.187
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
0.011 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
-1.081 -0.813 -0.634 -0.582 -0.625 -0.713 -0.848 -1.039
-0.036 -0.162 -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 52.0 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
47.8 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 49.7 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 53.3 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
```
### 10.3 Program Results

**nag_tsa_gain_phase_bivar (g13cfc) Example Program Results**

#### The gain

<table>
<thead>
<tr>
<th>Value</th>
<th>Lower bound</th>
<th>Upper bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3.1926</td>
<td>3.4685</td>
</tr>
<tr>
<td>1</td>
<td>3.1578</td>
<td>3.4384</td>
</tr>
<tr>
<td>2</td>
<td>3.0407</td>
<td>3.3018</td>
</tr>
<tr>
<td>3</td>
<td>2.7859</td>
<td>3.0178</td>
</tr>
<tr>
<td>4</td>
<td>2.6321</td>
<td>2.8954</td>
</tr>
<tr>
<td>5</td>
<td>2.4180</td>
<td>2.7416</td>
</tr>
<tr>
<td>6</td>
<td>2.1902</td>
<td>2.5372</td>
</tr>
<tr>
<td>7</td>
<td>2.1475</td>
<td>2.4917</td>
</tr>
<tr>
<td>8</td>
<td>2.0064</td>
<td>2.3677</td>
</tr>
<tr>
<td>9</td>
<td>1.8070</td>
<td>2.1665</td>
</tr>
<tr>
<td>10</td>
<td>1.7362</td>
<td>2.1362</td>
</tr>
<tr>
<td>11</td>
<td>1.5189</td>
<td>1.9531</td>
</tr>
<tr>
<td>12</td>
<td>1.4541</td>
<td>1.9269</td>
</tr>
<tr>
<td>13</td>
<td>1.3191</td>
<td>1.8461</td>
</tr>
<tr>
<td>14</td>
<td>1.0223</td>
<td>1.5720</td>
</tr>
<tr>
<td>15</td>
<td>0.9329</td>
<td>1.5070</td>
</tr>
<tr>
<td>16</td>
<td>0.8130</td>
<td>1.4183</td>
</tr>
<tr>
<td>17</td>
<td>0.6026</td>
<td>1.3448</td>
</tr>
<tr>
<td>18</td>
<td>0.2708</td>
<td>1.8485</td>
</tr>
<tr>
<td>19</td>
<td>0.3623</td>
<td>1.6367</td>
</tr>
<tr>
<td>20</td>
<td>0.3597</td>
<td>1.8653</td>
</tr>
<tr>
<td>21</td>
<td>0.3692</td>
<td>2.3027</td>
</tr>
<tr>
<td>22</td>
<td>0.3580</td>
<td>1.7095</td>
</tr>
<tr>
<td>23</td>
<td>0.2272</td>
<td>2.2012</td>
</tr>
<tr>
<td>24</td>
<td>0.1910</td>
<td>2.5243</td>
</tr>
<tr>
<td>25</td>
<td>0.3826</td>
<td>1.4128</td>
</tr>
<tr>
<td>26</td>
<td>0.4191</td>
<td>1.6261</td>
</tr>
<tr>
<td>27</td>
<td>0.2131</td>
<td>4.1606</td>
</tr>
<tr>
<td>28</td>
<td>0.2676</td>
<td>2.1656</td>
</tr>
<tr>
<td>29</td>
<td>0.3297</td>
<td>1.2512</td>
</tr>
<tr>
<td>30</td>
<td>0.4847</td>
<td>1.2585</td>
</tr>
<tr>
<td>31</td>
<td>0.3527</td>
<td>1.3882</td>
</tr>
<tr>
<td>32</td>
<td>0.1441</td>
<td>5.9677</td>
</tr>
<tr>
<td>33</td>
<td>0.0488</td>
<td>32316.6711</td>
</tr>
<tr>
<td>34</td>
<td>0.3765</td>
<td>2.0799</td>
</tr>
<tr>
<td>35</td>
<td>0.4432</td>
<td>1.7094</td>
</tr>
<tr>
<td>36</td>
<td>0.4336</td>
<td>1.7081</td>
</tr>
<tr>
<td>37</td>
<td>0.4064</td>
<td>1.6327</td>
</tr>
<tr>
<td>38</td>
<td>0.4199</td>
<td>2.5646</td>
</tr>
<tr>
<td>39</td>
<td>0.3529</td>
<td>6.8372</td>
</tr>
<tr>
<td>40</td>
<td>0.3865</td>
<td>6.5937</td>
</tr>
</tbody>
</table>

#### The phase

<table>
<thead>
<tr>
<th>Value</th>
<th>Lower bound</th>
<th>Upper bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3.1416</td>
<td>3.2245</td>
</tr>
<tr>
<td>1</td>
<td>3.4870</td>
<td>3.5722</td>
</tr>
<tr>
<td>2</td>
<td>3.8489</td>
<td>3.9313</td>
</tr>
<tr>
<td>3</td>
<td>4.2429</td>
<td>4.3228</td>
</tr>
<tr>
<td>4</td>
<td>4.5992</td>
<td>4.6946</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>5</td>
<td>5.0060</td>
<td>4.8805</td>
</tr>
<tr>
<td>6</td>
<td>5.4130</td>
<td>5.2659</td>
</tr>
<tr>
<td>7</td>
<td>5.7591</td>
<td>5.6104</td>
</tr>
<tr>
<td>8</td>
<td>6.0690</td>
<td>5.9034</td>
</tr>
<tr>
<td>9</td>
<td>0.0739</td>
<td>-0.1076</td>
</tr>
<tr>
<td>10</td>
<td>0.3419</td>
<td>0.1345</td>
</tr>
<tr>
<td>11</td>
<td>0.7480</td>
<td>0.4966</td>
</tr>
<tr>
<td>12</td>
<td>1.2763</td>
<td>0.9948</td>
</tr>
<tr>
<td>13</td>
<td>1.6757</td>
<td>1.3396</td>
</tr>
<tr>
<td>14</td>
<td>1.9390</td>
<td>1.5087</td>
</tr>
<tr>
<td>15</td>
<td>2.2669</td>
<td>1.7873</td>
</tr>
<tr>
<td>16</td>
<td>2.2803</td>
<td>1.7238</td>
</tr>
<tr>
<td>17</td>
<td>2.3027</td>
<td>1.5000</td>
</tr>
<tr>
<td>18</td>
<td>2.8645</td>
<td>0.9437</td>
</tr>
<tr>
<td>19</td>
<td>2.7577</td>
<td>1.2497</td>
</tr>
<tr>
<td>20</td>
<td>2.7583</td>
<td>1.1124</td>
</tr>
<tr>
<td>21</td>
<td>2.1779</td>
<td>0.3475</td>
</tr>
<tr>
<td>22</td>
<td>2.2593</td>
<td>0.6959</td>
</tr>
<tr>
<td>23</td>
<td>2.5261</td>
<td>0.2553</td>
</tr>
<tr>
<td>24</td>
<td>3.0780</td>
<td>0.4963</td>
</tr>
<tr>
<td>25</td>
<td>3.0754</td>
<td>1.7690</td>
</tr>
<tr>
<td>26</td>
<td>3.8081</td>
<td>2.4522</td>
</tr>
<tr>
<td>27</td>
<td>3.5382</td>
<td>0.5665</td>
</tr>
<tr>
<td>28</td>
<td>2.1386</td>
<td>0.0475</td>
</tr>
<tr>
<td>29</td>
<td>2.1469</td>
<td>0.8133</td>
</tr>
<tr>
<td>30</td>
<td>2.3176</td>
<td>1.3636</td>
</tr>
<tr>
<td>31</td>
<td>2.6071</td>
<td>1.2368</td>
</tr>
<tr>
<td>32</td>
<td>2.8184</td>
<td>-0.9052</td>
</tr>
<tr>
<td>33</td>
<td>3.2115</td>
<td>-10.1913</td>
</tr>
<tr>
<td>34</td>
<td>5.0543</td>
<td>3.3451</td>
</tr>
<tr>
<td>35</td>
<td>5.4770</td>
<td>4.1272</td>
</tr>
<tr>
<td>36</td>
<td>5.4155</td>
<td>4.0445</td>
</tr>
<tr>
<td>37</td>
<td>5.4122</td>
<td>4.0216</td>
</tr>
<tr>
<td>38</td>
<td>5.0105</td>
<td>3.2010</td>
</tr>
<tr>
<td>39</td>
<td>5.1738</td>
<td>2.2098</td>
</tr>
<tr>
<td>40</td>
<td>0.0000</td>
<td>-2.8369</td>
</tr>
</tbody>
</table>