NAG C Library Function Document

nag_complex_airy_bi (s17dhc)

1 Purpose

nag_complex_airy_bi (s17dhc) returns the value of the Airy function Bi(z) or its derivative Bi'(z) for complex z, with an option for exponential scaling.

2 Specification

void nag_complex_airy_bi (Nag_FunType deriv, Complex z, Nag_ScaleResType scal, Complex *bi, NagError *fail)

3 Description

nag_complex_airy_bi (s17dhc) returns a value for the Airy function Bi(z) or its derivative Bi'(z), where z is complex, \(-\pi < \arg z \leq \pi\). Optionally, the value is scaled by the factor \(e^{\left|\text{Re}(2z\sqrt{3}/3)\right|}\).

The function is derived from the routine CBIRY in Amos (1986). It is based on the relations

\[ Bi(z) = \frac{\sqrt{2}}{\sqrt{3}}(I_{1/3}(w) + I_{4/3}(w)), \]

and

\[ Bi'(z) = \frac{z}{\sqrt{3}}(I_{1/3}(w) + I_{4/3}(w)), \]

where \(I_0\) is the modified Bessel function and \(w = 2z\sqrt{3}/3\).

For very large \(|z|\), argument reduction will cause total loss of accuracy, and so no computation is performed. For slightly smaller \(|z|\), the computation is performed but results are accurate to less than half of machine precision. If \(\text{Re} z\) is too large, and the unscaled function is required, there is a risk of overflow and so no computation is performed. In all the above cases, a warning is given by the function.

4 References


5 Parameters

1: deriv – Nag_FunType

\(\text{Input}\)

\(\text{On entry:}\) specifies whether the function or its derivative is required.

If deriv = Nag_Function, \(Bi(z)\) is returned.

If deriv = Nag_Deriv, \(Bi'(z)\) is returned.

\(\text{Constraint:}\) deriv = Nag_Function or Nag_Deriv.

2: z – Complex

\(\text{Input}\)

\(\text{On entry:}\) the argument \(z\) of the function.

3: scal – Nag_ScaleResType

\(\text{Input}\)

\(\text{On entry:}\) the scaling option.

If scal = Nag_UnscaleRes, the result is returned unscaled.
If \( \text{scal} = \text{Nag ScaleRes} \), the result is returned scaled by the factor \( e^{\Re(2\sqrt{z^3}/3)} \).

Constraint: \( \text{scal} = \text{Nag UnscaleRes} \) or \( \text{Nag ScaleRes} \).

4: \( \text{bi} \) – Complex *

On exit: the required function or derivative value.

5: \( \text{fail} \) – NagError *

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

**NE_OVERFLOW_LIKELY**

No computation because \( z.\text{re} = \langle \text{value} \rangle \) is too large when \( \text{scal} = \text{Nag UnscaleRes} \).

**NE_TERMINATION_FAILURE**

No computation – algorithm termination condition not met.

**NE_TOTAL_PRECISION_LOSS**

No computation because \( \text{abs}(z) = \langle \text{value} \rangle > \langle \text{value} \rangle \).

**NW_SOME_PRECISION_LOSS**

Results lack precision because \( \text{abs}(z) = \langle \text{value} \rangle > \langle \text{value} \rangle \).

**NE_BAD_PARAM**

On entry, parameter \( \langle \text{value} \rangle \) had an illegal value.

**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 consult NAG for assistance.

7 Accuracy

All constants in \text{nag complex airy bi} (s17dhc) are given to approximately 18 digits of precision. Calling the number of digits of precision in the floating-point arithmetic being used \( t \), then clearly the maximum number of correct digits in the results obtained is limited by \( p = \min(t, 18) \). Because of errors in argument reduction when computing elementary functions inside \text{nag complex airy bi} (s17dhc), the actual number of correct digits is limited, in general, by \( p - s \), where \( s \approx \max(1, \lceil \log_{10} |z| \rceil) \) represents the number of digits lost due to the argument reduction. Thus the larger the value of \( |z| \), the less the precision in the result.

Empirical tests with modest values of \( z \), checking relations between Airy functions \( \text{Ai}(z), \text{Ai}'(z), \text{Bi}(z) \) and \( \text{Bi}'(z) \), have shown errors limited to the least significant 3 – 4 digits of precision.

8 Further Comments

Note that if the function is required to operate on a real argument only, then it may be much cheaper to call \text{nag airy bi} (s17ahc) or \text{nag airy bi deriv} (s17akc).

9 Example

The example program prints a caption and then proceeds to read sets of data from the input data stream. The first datum is a value for the parameter \text{deriv}, the second is a complex value for the argument, \( z \), and
the third is a character value used as a flag to set the parameter \texttt{scal}. The program calls the function and prints the results. The process is repeated until the end of the input data stream is encountered.

9.1 Program Text

```c
/* nag_complex_airy_bi (s17dhc) Example Program
 * Copyright 2002 Numerical Algorithms Group.
 * Mark 7, 2002.
 */

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

int main(void)
{
    Complex z, bi;
    Nag_ScaleResType scal_enum;
    Nag_FunType deriv_enum;
    char deriv, scal;
    Integer exit_status = EXIT_SUCCESS;
    NagError fail;

    INIT_FAIL(fail);
    /* Skip heading in data file */
    vscanf("%*[\n]");
    vprintf("s17dhc Example Program Results\n");
    vprintf(" deriv z scal bi\n");
    while (scanf(" '%c' (%lf,%lf) '%c'%*\[\n] ",
            &deriv, &z.re, &z.im, &scal) != EOF)
    {
        /* Convert scal character to enum */
        if (scal == 's')
        {
            scal_enum = Nag_ScaleRes;
        }
        else if (scal == 'u')
        {
            scal_enum = Nag_UnscaleRes;
        }
        else
        {
            vprintf("Unrecognised character for Nag_ScaleResType type\n");
            exit_status = -1;
            goto END;
        }
        /* Convert deriv character to enum */
        if (deriv == 'f')
        {
            deriv_enum = Nag_Function;
        }
        else if (deriv == 'd')
        {
            deriv_enum = Nag_Deriv;
        }
        else
        {
            vprintf("Unrecognised character for Nag_FunType type\n");
            exit_status = -1;
            goto END;
        }
        s17dhc(deriv_enum, z, scal_enum, &bi, &fail);
        if (fail.code == NE_NOERROR)
            vprintf(" '%c' (%7.3f,%7.3f) '%c' (%7.3f,%7.3f)\n",
                    deriv, z.re, z.im, scal, bi.re, bi.im);
    }
    ...
else {
    Vprintf("Error from s17dhc.\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
END:
return exit_status;
}

9.2 Program Data

s17dhc Example Program Data
'f' ( 0.3, 0.4) 'u'
'f' ( 0.2, 0.0) 'u'
'f' ( 1.1, -6.6) 'u'
'f' ( 1.1, -6.6) 's'
'd' (-1.0, 0.0) 'u' - Values of deriv, z and scal

9.3 Program Results

s17dhc Example Program Results
<table>
<thead>
<tr>
<th>deriv</th>
<th>z</th>
<th>scal</th>
<th>bi</th>
</tr>
</thead>
<tbody>
<tr>
<td>'f'</td>
<td>(0.300, 0.400) 'u'</td>
<td>(0.736, 0.183)</td>
<td></td>
</tr>
<tr>
<td>'f'</td>
<td>(0.200, 0.000) 'u'</td>
<td>(0.705, 0.000)</td>
<td></td>
</tr>
<tr>
<td>'f'</td>
<td>(1.100, -6.600) 'u'</td>
<td>(-47.904, 43.663)</td>
<td></td>
</tr>
<tr>
<td>'f'</td>
<td>(1.100, -6.600) 's'</td>
<td>(-0.130, 0.119)</td>
<td></td>
</tr>
<tr>
<td>'d'</td>
<td>(-1.000, 0.000) 'u'</td>
<td>(0.592, 0.000)</td>
<td></td>
</tr>
</tbody>
</table>