1 Purpose

nag_complex_airy_ai (s17dgc) returns the value of the Airy function \( Ai(z) \) or its derivative \( Ai'(z) \) for complex \( z \), with an option for exponential scaling.

2 Specification

```c
void nag_complex_airy_ai (Nag_FunType deriv, Complex z, Nag_ScaleResType scal,
                Complex *ai, Integer *nz, NagError *fail)
```

3 Description

nag_complex_airy_ai (s17dgc) returns a value for the Airy function \( Ai(z) \) or its derivative \( Ai'(z) \), where \( z \) is complex, \( -\pi < \text{arg} \, z \leq \pi \). Optionally, the value is scaled by the factor \( e^{2z\sqrt{3}/3} \).

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

\[
Ai(z) = \frac{\sqrt{2}K_{1/3}(w)}{\pi\sqrt{3}}, \quad \text{and} \quad Ai'(z) = -\frac{2zK_{2/3}(w)}{\pi\sqrt{3}},
\]

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

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 \( \text{Re} \, w \) 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: \( \text{deriv} \) – Nag_FunType

\( \text{Input} \)

On entry: specifies whether the function or its derivative is required.

If \( \text{deriv} = \text{Nag}\_\text{Function} \), \( Ai(z) \) is returned.

If \( \text{deriv} = \text{Nag}\_\text{Deriv} \), \( Ai'(z) \) is returned.

Constraint: \( \text{deriv} = \text{Nag}\_\text{Function} \) or \( \text{Nag}\_\text{Deriv} \).

2: \( z \) – Complex

\( \text{Input} \)

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

3: \( \text{scal} \) – Nag_ScaleResType

\( \text{Input} \)

On entry: the scaling option.

If \( \text{scal} = \text{Nag}\_\text{UnscaleRes} \), the result is returned unscaled.
If $\text{scal} = \text{Nag\_ScaleRes}$, the result is returned scaled by the factor $e^{\pi \sqrt{2} / 3}$.

$Constriant$: $\text{scal} = \text{Nag\_UnscaleRes}$ or $\text{Nag\_ScaleRes}$.

4: $a_i$ – Complex *  
   $Output$  
   $On\ exit$: the required function or derivative value.

5: $nz$ – Integer *  
   $Output$  
   $On\ exit$: $nz$ indicates whether or not $a_i$ is set to zero due to underflow. This can only occur when $\text{scal} = \text{Nag\_UnscaleRes}$. 
   If $nz = 0$, $a_i$ is not set to zero. 
   If $nz = 1$, $a_i$ is set to zero.

6: $\text{fail}$ – NagError *  
   $Input/Output$  
   The NAG error parameter (see the Essential Introduction).

6  Error Indicators and Warnings

$NE\_OVERFLOW\_LIKELY$

No computation because $\omega.\text{re}$ too large, where $\omega = (2/3) \times z^{3(2)}$.

$NE\_TERMINATION\_FAILURE$

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 nag_complex_airy_ai (s17dgc) 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 nag_complex_airy_ai (s17dgc), the actual number of correct digits is limited, in general, by $p - s$, where $s \approx \max(1, \log_{10}|z|)$ 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 nag_airy_ai (s17agc) or nag_airy_ai_deriv (s17ajc).

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

int main(void)
{
    Complex z, ai;
    Integer nz;
    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("s17dgc Example Program Results\n");
    Vprintf(" deriv z scal ai nz\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_Derivative;
        }
        else
        {
            Vprintf("Unrecognised character for Nag_FunType type\n");
            exit_status = -1;
            goto END;
        }
        /* Call the function */
        nag_complex_airy_ai(scal_enum, deriv_enum, z, &ai);
        /* Print the results */
        Vprintf(" deriv z scal ai nz\n");
    }
}
```

[NP3645/7]
deriv_enum = Nag_Deriv;
}
else
{
    Vprintf("Unrecognised character for Nag_FunType type\n");
    exit_status = -1;
    goto END;
}
s17dgc(deriv_enum, z, scal_enum, &ai, &nz, &fail);
if (fail.code == NE_NOERROR)
    Vprintf(" '%c' (%7.3f,%7.3f) '%c' (%7.3f,%7.3f) %ld\n",
            deriv, z.re, z.im, scal, ai.re, ai.im, nz);
else
{
    Vprintf("Error from s17dgc.\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
}
END:
return exit_status;
}

9.2 Program Data

s17dgc 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

s17dgc Example Program Results

<table>
<thead>
<tr>
<th>deriv</th>
<th>z</th>
<th>scal</th>
<th>ai</th>
<th>nz</th>
</tr>
</thead>
<tbody>
<tr>
<td>'f'</td>
<td>( 0.300, 0.400)</td>
<td>'u'</td>
<td>( 0.272, -0.100)</td>
<td>0</td>
</tr>
<tr>
<td>'f'</td>
<td>( 0.200, 0.000)</td>
<td>'u'</td>
<td>( 0.304, 0.000)</td>
<td>0</td>
</tr>
<tr>
<td>'f'</td>
<td>( 1.100, -6.600)</td>
<td>'u'</td>
<td>(-43.663, -47.903)</td>
<td>0</td>
</tr>
<tr>
<td>'f'</td>
<td>( 1.100, -6.600)</td>
<td>'s'</td>
<td>( 0.165, 0.060)</td>
<td>0</td>
</tr>
<tr>
<td>'d'</td>
<td>(-1.000, 0.000)</td>
<td>'u'</td>
<td>(-0.010, 0.000)</td>
<td>0</td>
</tr>
</tbody>
</table>