

NAG C Library Function Document

nag_complex_airy_ai (s17dgc)

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

```
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, $-\pi < \arg z \leq \pi$. Optionally, the value is scaled by the factor $e^{2z\sqrt{z}/3}$.

The function is derived from the routine CAIRY in Amos (1986). It is based on the relations $Ai(z) = \frac{\sqrt{z}K_{1/3}(w)}{\pi\sqrt{3}}$, and $Ai'(z) = \frac{-zK_{2/3}(w)}{\pi\sqrt{3}}$, where K_ν is the modified Bessel function and $w = 2z\sqrt{z}/3$.

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

Abramowitz M and Stegun I A (1972) *Handbook of Mathematical Functions* (3rd Edition) Dover Publications

Amos D E (1986) Algorithm 644: A portable package for Bessel functions of a complex argument and non-negative order *ACM Trans. Math. Software* **12** 265–273

5 Parameters

- 1: **deriv** – Nag_FunType *Input*
On entry: specifies whether the function or its derivative is required.
 If **deriv** = **Nag_Function**, $Ai(z)$ is returned.
 If **deriv** = **Nag_Deriv**, $Ai'(z)$ is returned.
Constraint: **deriv** = **Nag_Function** or **Nag_Deriv**.
- 2: **z** – Complex *Input*
On entry: the argument z of the function.
- 3: **scal** – Nag_ScaleResType *Input*
On entry: the scaling option.
 If **scal** = **Nag_UnscaleRes**, the result is returned unscaled.

If **scal** = **Nag_ScaleRes**, the result is returned scaled by the factor $e^{2z\sqrt{z}/3}$.

Constraint: **scal** = **Nag_UnscaleRes** or **Nag_ScaleRes**.

- 4: **ai** – Complex * *Output*
 On exit: the required function or derivative value.
- 5: **nz** – Integer * *Output*
 On exit: **nz** indicates whether or not **ai** is set to zero due to underflow. This can only occur when **scal** = **Nag_UnscaleRes**.
 If **nz** = 0, **ai** is not set to zero.
 If **nz** = 1, **ai** is set to zero.
- 6: **fail** – NagError * *Input/Output*
 The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_OVERFLOW_LIKELY

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

NE_TERMINATION_FAILURE

No computation – algorithm termination condition not met.

NE_TOTAL_PRECISION_LOSS

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

NW_SOME_PRECISION_LOSS

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

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_complexairy_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_complexairy_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

```
/* nag_complex_airy_ai (s17dgc) 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, 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_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

deriv	z	scal	ai	nz
'f'	(0.300, 0.400)	'u'	(0.272, -0.100)	0
'f'	(0.200, 0.000)	'u'	(0.304, 0.000)	0
'f'	(1.100, -6.600)	'u'	(-43.663, -47.903)	0
'f'	(1.100, -6.600)	's'	(0.165, 0.060)	0
'd'	(-1.000, 0.000)	'u'	(-0.010, 0.000)	0
