NAG Library Function Document

nag_complex_airy_ai (s17dgc)

1 Purpose

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

2 Specification

```
#include <nag.h>
#include <nags.h>
```

```
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 $\mathrm{Ai}(z)$ or its derivative $\mathrm{Ai}'(z)$, 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 function CAIRY in Amos (1986). It is based on the relations $\mathrm{Ai}(z) = \dfrac{\sqrt{z}K_{1/3}(w)}{\pi\sqrt{3}}$, and $\mathrm{Ai}'(z) = \dfrac{-zK_{2/3}(w)}{\pi\sqrt{3}}$, 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 $\mathrm{Re}(w)$ 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 Arguments

1: **deriv** – Nag_FunType *Input*

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

deriv = Nag_Function
$\mathrm{Ai}(z)$ is returned.

deriv = Nag_Deriv
$\mathrm{Ai}'(z)$ 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.

scal = Nag_UnscaleRes
The result is returned unscaled.

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: indicates whether or not **ai** is set to zero due to underflow. This can only occur when **scal** = Nag_UnscaleRes.

nz = 0
ai is not set to zero.

nz = 1
ai is set to zero.

6: **fail** – NagError * *Input/Output*

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

6 Error Indicators and Warnings

NE_BAD_PARAM

On entry, argument $\langle\mathit{value}\rangle$ 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 contact NAG for assistance.

NE_OVERFLOW_LIKELY

No computation because $\omega.re$ 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 $|\mathbf{z}| = \langle\mathit{value}\rangle > \langle\mathit{value}\rangle$.

NW_SOME_PRECISION_LOSS

Results lack precision because $|\mathbf{z}| = \langle\mathit{value}\rangle > \langle\mathit{value}\rangle$.

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 $\mathrm{Ai}(z)$, $\mathrm{Ai}'(z)$, $\mathrm{Bi}(z)$ and $\mathrm{Bi}'(z)$, have shown errors limited to the least significant 3 – 4 digits of precision.

8 Parallelism and Performance

Not applicable.

9 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).

10 Example

This example prints a caption and then proceeds to read sets of data from the input data stream. The first datum is a value for the argument **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 argument **scal**. The program calls the function and prints the results. The process is repeated until the end of the input data stream is encountered.

10.1 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)
{
  Integer           exit_status = 0;
  Complex           z, ai;
  Integer           nz;
  char              nag_enum_deriv[40], nag_enum_scal[40];
  Nag_ScaleResType scal;
  Nag_FunType       deriv;
  NagError          fail;

  INIT_FAIL(fail);

  /* Skip heading in data file */
  scanf("%*[^\n]");
  printf("nag_complex_airy_ai (s17dgc) Example Program Results\n");
  printf("    deriv               z              scal                 "
          "ai        nz\n");
  while (scanf(" %39s (%lf,%lf) %39s%*[^\n] ",
               nag_enum_deriv, &z.re, &z.im, nag_enum_scal) != EOF)
    {
      /* nag_enum_name_to_value (x04nac).
       * Converts NAG enum member name to value
       */
      deriv = (Nag_FunType) nag_enum_name_to_value(nag_enum_deriv);
      scal = (Nag_ScaleResType) nag_enum_name_to_value(nag_enum_scal);

      /* nag_complex_airy_ai (s17dgc).
       * Airy functions Ai(z), complex z
       */
      nag_complex_airy_ai(deriv, z, scal, &ai, &nz, &fail);
      if (fail.code != NE_NOERROR)
        {
          printf("Error from nag_complex_airy_ai (s17dgc).\n%s\n",
                  fail.message);
          exit_status = 1;
          goto END;
        }
      printf(" %-12s  (%7.3f,%7.3f)  %-14s  (%7.3f,%7.3f)  %ld\n",
              nag_enum_deriv, z.re, z.im, nag_enum_scal, ai.re, ai.im, nz);
    }

 END:

  return exit_status;
}
```

10.2 Program Data

```
nag_complex_airy_ai (s17dgc) Example Program Data
 Nag_Function   ( 0.3,  0.4)    Nag_UnscaleRes
 Nag_Function   ( 0.2,  0.0)    Nag_UnscaleRes
 Nag_Function   ( 1.1, -6.6)    Nag_UnscaleRes
 Nag_Function   ( 1.1, -6.6)    Nag_ScaleRes
 Nag_Deriv      (-1.0,  0.0)    Nag_UnscaleRes   - Values of deriv, z and scal
```

10.3 Program Results

```
nag_complex_airy_ai (s17dgc) Example Program Results
    deriv              z              scal              ai          nz
 Nag_Function  (  0.300,  0.400)  Nag_UnscaleRes  (  0.272, -0.100)  0
 Nag_Function  (  0.200,  0.000)  Nag_UnscaleRes  (  0.304,  0.000)  0
 Nag_Function  (  1.100, -6.600)  Nag_UnscaleRes  (-43.663,-47.903)  0
 Nag_Function  (  1.100, -6.600)  Nag_ScaleRes    (  0.165,  0.060)  0
 Nag_Deriv     ( -1.000,  0.000)  Nag_UnscaleRes  ( -0.010,  0.000)  0
```