NAG Library Function Document

nag_airy_ai_vector (s17auc)

1 Purpose

nag_airy_ai_vector (s17auc) returns an array of values for the Airy function, $\mathrm{Ai}(x)$.

2 Specification

```
#include <nag.h>
#include <nags.h>
void nag_airy_ai_vector (Integer n, const double x[], double f[],
     Integer ivalid[], NagError *fail)
```

3 Description

nag_airy_ai_vector (s17auc) evaluates an approximation to the Airy function, $\mathrm{Ai}(x_i)$ for an array of arguments x_i, for $i = 1, 2, \ldots, n$. It is based on a number of Chebyshev expansions:

For $x < -5$,

$$\mathrm{Ai}(x) = \frac{a(t)\sin z - b(t)\cos z}{(-x)^{1/4}}$$

where $z = \frac{\pi}{4} + \frac{2}{3}\sqrt{-x^3}$, and $a(t)$ and $b(t)$ are expansions in the variable $t = -2\left(\frac{5}{x}\right)^3 - 1$.

For $-5 \leq x \leq 0$,

$$\mathrm{Ai}(x) = f(t) - xg(t),$$

where f and g are expansions in $t = -2\left(\frac{x}{5}\right)^3 - 1$.

For $0 < x < 4.5$,

$$\mathrm{Ai}(x) = e^{-3x/2}y(t),$$

where y is an expansion in $t = 4x/9 - 1$.

For $4.5 \leq x < 9$,

$$\mathrm{Ai}(x) = e^{-5x/2}u(t),$$

where u is an expansion in $t = 4x/9 - 3$.

For $x \geq 9$,

$$\mathrm{Ai}(x) = \frac{e^{-z}v(t)}{x^{1/4}},$$

where $z = \frac{2}{3}\sqrt{x^3}$ and v is an expansion in $t = 2\left(\frac{18}{z}\right) - 1$.

For $|x| <$ ***machine precision***, the result is set directly to $\mathrm{Ai}(0)$. This both saves time and guards against underflow in intermediate calculations.

For large negative arguments, it becomes impossible to calculate the phase of the oscillatory function with any precision and so the function must fail. This occurs if $x < -\left(\frac{3}{2\epsilon}\right)^{2/3}$, where ϵ is the ***machine precision***.

For large positive arguments, where Ai decays in an essentially exponential manner, there is a danger of underflow so the function must fail.

4 References

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

5 Arguments

1: **n** – Integer *Input*

On entry: n, the number of points.

Constraint: **n** ≥ 0.

2: **x**[**n**] – const double *Input*

On entry: the argument x_i of the function, for $i = 1, 2, \ldots,$ **n**.

3: **f**[**n**] – double *Output*

On exit: $\mathrm{Ai}(x_i)$, the function values.

4: **ivalid**[**n**] – Integer *Output*

On exit: **ivalid**$[i-1]$ contains the error code for x_i, for $i = 1, 2, \ldots,$ **n**.

ivalid$[i-1] = 0$
No error.

ivalid$[i-1] = 1$
x_i is too large and positive. **f**$[i-1]$ contains zero. The threshold value is the same as for **fail.code** = NE_REAL_ARG_GT in nag_airy_ai (s17agc), as defined in the Users' Note for your implementation.

ivalid$[i-1] = 2$
x_i is too large and negative. **f**$[i-1]$ contains zero. The threshold value is the same as for **fail.code** = NE_REAL_ARG_LT in nag_airy_ai (s17agc), as defined in the Users' Note for your implementation.

5: **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 ⟨*value*⟩ had an illegal value.

NE_INT

On entry, **n** = ⟨*value*⟩.
Constraint: **n** ≥ 0.

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.

NW_IVALID

On entry, at least one value of **x** was invalid.
Check **ivalid** for more information.

7 Accuracy

For negative arguments the function is oscillatory and hence absolute error is the appropriate measure. In the positive region the function is essentially exponential-like and here relative error is appropriate. The absolute error, E, and the relative error, ϵ, are related in principle to the relative error in the argument, δ, by

$$E \simeq |x\mathrm{Ai}'(x)|\delta, \epsilon \simeq \left|\frac{x\mathrm{Ai}'(x)}{\mathrm{Ai}(x)}\right|\delta.$$

In practice, approximate equality is the best that can be expected. When δ, ϵ or E is of the order of the ***machine precision***, the errors in the result will be somewhat larger.

For small x, errors are strongly damped by the function and hence will be bounded by the ***machine precision***.

For moderate negative x, the error behaviour is oscillatory but the amplitude of the error grows like

$$\text{amplitude}\left(\frac{E}{\delta}\right) \sim \frac{|x|^{5/4}}{\sqrt{\pi}}.$$

However the phase error will be growing roughly like $\frac{2}{3}\sqrt{|x|^3}$ and hence all accuracy will be lost for large negative arguments due to the impossibility of calculating sin and cos to any accuracy if $\frac{2}{3}\sqrt{|x|^3} > \frac{1}{\delta}$.

For large positive arguments, the relative error amplification is considerable:

$$\frac{\epsilon}{\delta} \sim \sqrt{x^3}.$$

This means a loss of roughly two decimal places accuracy for arguments in the region of 20. However very large arguments are not possible due to the danger of setting underflow and so the errors are limited in practice.

8 Parallelism and Performance

Not applicable.

9 Further Comments

None.

10 Example

This example reads values of **x** from a file, evaluates the function at each value of x_i and prints the results.

10.1 Program Text

```
/* nag_airy_ai_vector (s17auc) Example Program.
 *
 * Copyright 2011, Numerical Algorithms Group.
 *
 * Mark 23 2011.
 */
#include <nag.h>
```

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

int main(void)
{
  Integer  exit_status = 0;
  Integer  i, n;
  double   *f = 0, *x = 0;
  Integer  *ivalid = 0;
  NagError fail;

  INIT_FAIL(fail);

  /* Skip heading in data file */
  scanf("%*[^\n]");

  printf("nag_airy_ai_vector (s17auc) Example Program Results\n");
  printf("\n");
  printf("     x           f          ivalid\n");
  printf("\n");
  scanf("%ld", &n);
  scanf("%*[^\n]");

  /* Allocate memory */
  if (!(x = NAG_ALLOC(n, double)) ||
      !(f = NAG_ALLOC(n, double)) ||
      !(ivalid = NAG_ALLOC(n, Integer)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  for (i=0; i<n; i++)
    scanf("%lf", &x[i]);
  scanf("%*[^\n]");

  /* nag_airy_ai_vector (s17auc).
   * Airy function Ai(x)
   */
  nag_airy_ai_vector(n, x, f, ivalid, &fail);
  if (fail.code!=NE_NOERROR && fail.code!=NW_IVALID)
    {
      printf("Error from nag_airy_ai_vector (s17auc).\n%s\n",
              fail.message);
      exit_status = 1;
      goto END;
    }

  for (i=0; i<n; i++)
    printf(" %11.3e %11.3e %4ld\n", x[i], f[i], ivalid[i]);

 END:
  NAG_FREE(f);
  NAG_FREE(x);
  NAG_FREE(ivalid);

  return exit_status;
}
```

10.2 Program Data

```
nag_airy_ai_vector (s17auc) Example Program Data

7

-10.0 -1.0 0.0 1.0 5.0 10.0 20.0
```

10.3 Program Results

```
nag_airy_ai_vector (s17auc) Example Program Results

     x           f          ivalid

  -1.000e+01   4.024e-02    0
  -1.000e+00   5.356e-01    0
   0.000e+00   3.550e-01    0
   1.000e+00   1.353e-01    0
   5.000e+00   1.083e-04    0
   1.000e+01   1.105e-10    0
   2.000e+01   1.692e-27    0
```