NAG Library Function Document

nag_airy_bi_deriv_vector (s17axc)

1 Purpose

nag_airy_bi_deriv_vector (s17axc) returns an array of values for the derivative of the Airy function $\mathrm{Bi}(x)$.

2 Specification

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

```
void nag_airy_bi_deriv_vector (Integer n, const double x[], double f[],
     Integer ivalid[], NagError *fail)
```

3 Description

nag_airy_bi_deriv_vector (s17axc) calculates an approximate value for the derivative of the Airy function $\mathrm{Bi}(x_i)$ for an array of arguments x_i, for $i = 1, 2, \dots, n$. It is based on a number of Chebyshev expansions.

For $x < -5$,

$$\mathrm{Bi}'(x) = \sqrt[4]{-x}\left[-a(t)\sin z + \frac{b(t)}{\zeta}\cos z\right],$$

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

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

$$\mathrm{Bi}'(x) = \sqrt{3}\left(x^2 f(t) + g(t)\right),$$

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

For $0 < x < 4.5$,

$$\mathrm{Bi}'(x) = e^{3x/2} y(t),$$

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

For $4.5 \le x < 9$,

$$\mathrm{Bi}'(x) = e^{21x/8} u(t),$$

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

For $x \ge 9$,

$$\mathrm{Bi}'(x) = \sqrt[4]{x} e^{z} v(t),$$

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

For $|x| <$ the square of the ***machine precision***, the result is set directly to $\mathrm{Bi}'(0)$. This saves time and avoids possible underflows in calculation.

For large negative arguments, it becomes impossible to calculate a result for the oscillating function with any accuracy so the function must fail. This occurs for $x < -\left(\frac{\sqrt{\pi}}{\epsilon}\right)^{4/7}$, where ϵ is the ***machine precision***.

For large positive arguments, where Bi' grows in an essentially exponential manner, there is a danger of overflow 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: $\mathbf{n} \geq 0$.

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

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

3: **f**[**n**] – double *Output*

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

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

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

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_bi_deriv (s17akc), 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_bi_deriv (s17akc), as defined in the Users' Note for your implementation.

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

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 3.2.1.2 in How to Use the NAG Library and its Documentation for further information.

NE_BAD_PARAM

On entry, argument ⟨*value*⟩ had an illegal value.

NE_INT

On entry, **n** $= \langle value \rangle$.
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.

An unexpected error has been triggered by this function. Please contact NAG.
See Section 3.6.6 in How to Use the NAG Library and its Documentation for further information.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.
See Section 3.6.5 in How to Use the NAG Library and its Documentation for further information.

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 appropriate. In the positive region the function has essentially exponential behaviour and hence relative error is needed. The absolute error, E, and the relative error ϵ, are related in principle to the relative error in the argument δ, by

$$E \simeq \left|x^2 \operatorname{Bi}(x)\right|\delta \quad \epsilon \simeq \left|\frac{x^2 \operatorname{Bi}(x)}{\operatorname{Bi}'(x)}\right|\delta.$$

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, positive or negative, errors are strongly attenuated by the function and hence will effectively be bounded by the ***machine precision***.

For moderate to large negative x, the error is, like the function, oscillatory. However, the amplitude of the absolute error grows like $\frac{|x|^{7/4}}{\sqrt{\pi}}$. Therefore it becomes impossible to calculate the function with any accuracy if $|x|^{7/4} > \frac{\sqrt{\pi}}{\delta}$.

For large positive x, the relative error amplification is considerable: $\frac{\epsilon}{\delta} \sim \sqrt{x^3}$. However, very large arguments are not possible due to the danger of overflow. Thus in practice the actual amplification that occurs is limited.

8 Parallelism and Performance

nag_airy_bi_deriv_vector (s17axc) is not threaded in any implementation.

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_bi_deriv_vector (s17axc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */
#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 */
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif

  printf("nag_airy_bi_deriv_vector (s17axc) Example Program Results\n");
  printf("\n");
  printf("     x           f          ivalid\n");
  printf("\n");
#ifdef _WIN32
  scanf_s("%" NAG_IFMT "", &n);
#else
  scanf("%" NAG_IFMT "", &n);
#endif
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif

  /* 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++)
#ifdef _WIN32
    scanf_s("%lf", &x[i]);
#else
    scanf("%lf", &x[i]);
#endif
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif

  /* nag_airy_bi_deriv_vector (s17axc).
   * Derivative of the Airy function Bi(x)
   */
```

```
  nag_airy_bi_deriv_vector(n, x, f, ivalid, &fail);
  if (fail.code != NE_NOERROR && fail.code != NW_IVALID) {
    printf("Error from nag_airy_bi_deriv_vector (s17axc).\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
  }

  for (i = 0; i < n; i++)
    printf(" %11.3e %11.3e %4" NAG_IFMT "\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_bi_deriv_vector (s17axc) Example Program Data

7

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

10.3 Program Results

```
nag_airy_bi_deriv_vector (s17axc) Example Program Results

     x           f          ivalid

  -1.000e+01   1.194e-01    0
  -1.000e+00   5.924e-01    0
   0.000e+00   4.483e-01    0
   1.000e+00   9.324e-01    0
   5.000e+00   1.436e+03    0
   1.000e+01   1.429e+09    0
   2.000e+01   9.382e+25    0
```