NAG Library Function Document

\textbf{nag_airy_bi_vector (s17avc)}

1 Purpose

\texttt{nag_airy_bi_vector (s17avc)} returns an array of values of the Airy function, Bi(x).

2 Specification

\begin{verbatim}
#include <nag.h>
#include <nags.h>

void nag_airy_bi_vector (Integer n, const double x[], double f[],
                        Integer ivalid[], NagError *fail)
\end{verbatim}

3 Description

\texttt{nag_airy_bi_vector (s17avc)} evaluates an approximation to the Airy function Bi(x) for an array of arguments \(x_i\), for \(i = 1, 2, \ldots, n\). It is based on a number of Chebyshev expansions.

For \(x < -5\),

\[
Bi(x) = \frac{a(t) \cos z + b(t) \sin z}{(-x)^{1/4}},
\]

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 \(-5 \leq x \leq 0\),

\[
Bi(x) = \sqrt{3}(f(t) + xg(t)),
\]

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

For \(0 < x < 4.5\),

\[
Bi(x) = e^{11x/8}y(t),
\]

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

For \(4.5 \leq x < 9\),

\[
Bi(x) = e^{5x/2}v(t),
\]

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

For \(x \geq 9\),

\[
Bi(x) = e^{z^2/4}u(t),
\]

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

For \(|x| < \text{machine precision}\), the result is set directly to Bi(0). This both saves time and avoids possible intermediate underflows.
For large negative arguments, it becomes impossible to calculate the phase of the oscillating function with any accuracy so the function must fail. This occurs if \( x < -\left(\frac{3}{2\epsilon}\right)^{2/3} \), where \( \epsilon \) is the machine precision.

For large positive arguments, there is a danger of causing overflow since \( B_i \) grows in an essentially exponential manner, so the function must fail.

4 References


5 Arguments

1: \( n \) – Integer 
   \( \text{Input} \)
   On entry: \( n \), the number of points.
   Constraint: \( n \geq 0 \).

2: \( x[n] \) – const double 
   \( \text{Input} \)
   On entry: the argument \( x_i \) of the function, for \( i = 1, 2, \ldots, n \).

3: \( f[n] \) – double 
   \( \text{Output} \)
   On exit: \( B_i(x_i) \), the function values.

4: \( \text{ivalid}[n] \) – Integer 
   \( \text{Output} \)
   On exit: \( \text{ivalid}[i - 1] \) contains the error code for \( x_i \), for \( i = 1, 2, \ldots, n \).

   \( \text{ivalid}[i - 1] = 0 \)
   No error.

   \( \text{ivalid}[i - 1] = 1 \)
   \( x_i \) is too large and positive. \( f[i - 1] \) contains zero. The threshold value is the same as for \( \text{fail.code} = \text{NE_REAL_ARG_GT in nag_airy_bi (s17ahc)} \), as defined in the Users’ Note for your implementation.

   \( \text{ivalid}[i - 1] = 2 \)
   \( x_i \) is too large and negative. \( f[i - 1] \) contains zero. The threshold value is the same as for \( \text{fail.code} = \text{NE_REAL_ARG_LT in nag_airy_bi (s17ahc)} \), as defined in the Users’ Note for your implementation.

5: \( \text{fail} \) – NagError * 
   \( \text{Input/Output} \)
   The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

\( \text{NE_ALLOC_FAIL} \)
Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

\( \text{NE_BAD_PARAM} \)
On entry, argument \( \langle \text{value} \rangle \) had an illegal value.
On entry, $n = \langle \text{value} \rangle$.
Constraint: $n \geq 0$.

**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 the Essential Introduction 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 the Essential Introduction for further information.

**NW_IVALID**
On entry, at least one value of $x$ was invalid.
Check invalid 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, $\epsilon$, are related in principle to the relative error in the argument, $\delta$, by

$$E \approx |xB'i(x)|\delta, \quad \epsilon \approx \left| \frac{xB'i(x)}{Bi(x)} \right| \delta.$$  

In practice, approximate equality is the best that can be expected. When $\delta, \epsilon$ 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 and hence will be bounded essentially by the **machine precision**. For moderate to large negative $x$, the error behaviour is clearly oscillatory but the amplitude of the error grows like amplitude $\left( \frac{E}{\delta} \right) \sim \frac{|x|^{5/4}}{\sqrt{\pi}}$.

However the phase error will be growing roughly as $\frac{2}{3} \sqrt{|x|^3}$ and hence all accuracy will be lost for large negative arguments. This is 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 causing overflow and errors are therefore 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

```c
/* nag_airy_bi_vector (s17avc) Example Program. *
 * Copyright 2014 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 */
    #ifdef _WIN32
    scanf_s("%*[\n]");
    #else
    scanf("%*[\n]");
    #endif
    printf("nag_airy_bi_vector (s17avc) 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++)
    {
        if (i >= n)
            break;
        scanf("%lf", &x[i]);
        scanf("%lf", &f[i]);
    }
    goto END;
}
```

NAG Library Manual

s17avc

Mark 25
scanf("%*[\`\n]"vell
#endif

/* nag_airy_bi_vector (s17avc).
   * Airy function Bi(x)
   */

nag_airy_bi_vector(n, x, f, ivalid, &fail);
if (fail.code!=NE_NOERROR && fail.code!=NW_IVALID)
{
    printf("Error from nag_airy_bi_vector (s17avc).\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_vector (s17avc) 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_vector (s17avc) Example Program Results

<table>
<thead>
<tr>
<th>x</th>
<th>f</th>
<th>ivalid</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1.000e+01</td>
<td>-3.147e-01</td>
<td>0</td>
</tr>
<tr>
<td>-1.000e+00</td>
<td>1.040e-01</td>
<td>0</td>
</tr>
<tr>
<td>0.000e+00</td>
<td>6.149e-01</td>
<td>0</td>
</tr>
<tr>
<td>1.000e+00</td>
<td>1.207e+00</td>
<td>0</td>
</tr>
<tr>
<td>5.000e+00</td>
<td>6.578e+02</td>
<td>0</td>
</tr>
<tr>
<td>1.000e+01</td>
<td>4.556e+08</td>
<td>0</td>
</tr>
<tr>
<td>2.000e+01</td>
<td>2.104e+25</td>
<td>0</td>
</tr>
</tbody>
</table>