NAG Library Function Document

nag_bessel_i_alpha (s18ejc)

1 Purpose

nag_bessel_i_alpha (s18ejc) returns a sequence of values for the modified Bessel functions $I_{\alpha+n-1}(x)$ or $I_{\alpha+n+1}(x)$ for real $x$, non-negative $\alpha < 1$ and $n = 1, 2, \ldots, |N| + 1$.

2 Specification

```c
#include <nag.h>
#include <nags.h>
void nag_bessel_i_alpha (double x, double a, Integer nl, Complex b[],
   NagError *fail)
```

3 Description

nag_bessel_i_alpha (s18ejc) evaluates a sequence of values for the modified Bessel function of the first kind $I_{\alpha}(x)$, where $x$ is real and nonzero and $\alpha$ is the order with $0 \leq \alpha < 1$. The ($|N| + 1$)-member sequence is generated for orders $\alpha, \alpha + 1, \ldots, \alpha + N$ when $N \geq 0$. Note that $+$ is replaced by $-$ when $N < 0$. For positive orders the function may also be called with $x = 0$, since $I_q(0) = 0$ when $q > 0$. For negative orders the formula

$$I_{-q}(x) = I_q(x) + \frac{2}{\pi} \sin(\pi q) K_q(x)$$

is used to generate the required sequence.

4 References


5 Arguments

1: x – double

   On entry: the argument $x$ of the function.
   Constraint: if nl < 0, $x \neq 0.0$.

2: a – double

   On entry: the order $\alpha$ of the first member in the required sequence of function values.
   Constraint: $0.0 \leq a < 1.0$.

3: nl – Integer

   On entry: the value of $N$.
   Constraint: abs(nl) \leq 101.

4: b[x] – Complex

   On exit: with fail.code = NE_NOERROR or fail.code = NW_SOME_PRECISION_LOSS, the required sequence of function values: $b(n)$ contains $I_{\alpha+n-1}(x)$ if nl \geq 1 and $I_{\alpha+n+1}(x)$ otherwise, for $n = 1, 2, \ldots, abs(nl) + 1$. 

Mark 25
5:  fail – NagError *

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

6  Error Indicators and Warnings

NE_INT

On entry, nl = \langle value\rangle.
Constraint: abs(nl) \leq 101.

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

The evaluation has been abandoned due to the likelihood of overflow.

NE_REAL

On entry, a = \langle value\rangle.
Constraint: 0 : 0 : a < 1 : 0.

NE_REAL_INT

On entry, x = \langle value\rangle, nl = \langle value\rangle.
Constraint: x \neq 0 : 0 when nl < 0.

NE_TERMINATION_FAILURE

The evaluation has been abandoned due to failure to satisfy the termination condition.

NE_TOTAL_PRECISION_LOSS

The evaluation has been abandoned due to total loss of precision.

NW_SOME_PRECISION_LOSS

The evaluation has been completed but some precision has been lost.

7  Accuracy

All constants in the underlying functions are specified to approximately 18 digits of precision. If t denotes the number of digits of precision in the floating-point arithmetic being used, 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 the underlying functions, the actual number of correct digits is limited, in general, by \( p - s \), where \( s \approx \max(1, |\log_{10}|x||, |\log_{10}|\alpha||) \) represents the number of digits lost due to the argument reduction. Thus the larger the values of \(|x|\) and \(|\alpha|\), the less the precision in the result.

8  Parallelism and Performance

Not applicable.

9  Further Comments

None.
10 Example

The example program evaluates $I_0(x), I_1(x), I_2(x)$ and $I_3(x)$ at $x = 0.5$, and prints the results.

10.1 Program Text

```c
#include <stdio.h>
#include <nag.h>
#include <naq_stdlib.h>
#include <nags.h>

int main(void)
{
    Complex *b = 0;
    Integer exit_status = 0, i, nl;
    NagError fail;
    double a, alpha, d, x;
    INIT_FAIL(fail);

    /* Skip heading in data file */
    #ifdef _WIN32
        scanf_s("%*[\n]");
    #else
        scanf("%*[\n]");
    #endif
    printf("nag_bessel_i_alpha (s18ejc) Example Program Results
");
    if (!(b = NAG_ALLOC(101, Complex)))
    {
        printf("Allocation failure
");
        exit_status = -1;
        goto END;
    }

    /* nag_bessel_i_alpha (s18ejc).
     * Modified Bessel functions $I_{\alpha+n-1}(x)$ or
     * $I_{\alpha-n+1}(x)$ for real $x \neq 0$, non-negative
     * $\alpha < 1$ and $n = 1,2,...,|N|+1$
     */
    nag_bessel_i_alpha(x, a, nl, b, &fail);
    if (fail.code == NE_NOERROR)
    {
        printf(" Requested values of I_\alpha(X)

");
        alpha = a;
        printf(" alpha I_\alpha(X)
");
        for (i = 1; i <= ABS(nl) + 1; ++i)
        {
            printf("%13.4e (%13.4e, %13.4e)
", alpha, b[i - 1].re, b[i - 1].im);
            d = (double) nl;
            alpha += SIGN(1.0, d);
        }
    }
}
```
else
  {
    printf("Error from nag_bessel_i_alpha (s18ejc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }
END:
NAG_FREE(b);
return exit_status;
} /* main */

10.2 Program Data

nag_bessel_i_alpha (s18ejc) Example Program Data
0.5  0.0  3 : Values of x, a and nl

10.3 Program Results

nag_bessel_i_alpha (s18ejc) Example Program Results
x    a    nl
0.5  0.0  3

Requested values of I_alpha(X)

<table>
<thead>
<tr>
<th>alpha</th>
<th>I_alpha(X)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0000e+00</td>
<td>(1.0635e+00,  0.0000e+00)</td>
</tr>
<tr>
<td>1.0000e+00</td>
<td>(2.5789e-01,  0.0000e+00)</td>
</tr>
<tr>
<td>2.0000e+00</td>
<td>(3.1906e-02,  0.0000e+00)</td>
</tr>
<tr>
<td>3.0000e+00</td>
<td>(2.6451e-03,  0.0000e+00)</td>
</tr>
</tbody>
</table>