NAG C 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

```
void nag_bessel_i_alpha (double x, double a, Integer nl, Complex b,
          NagError *fail)
```

3 Description

This routine evaluates a sequence of values for the modified Bessel function of the first kind $I_\alpha(x)$, where x is real and non-zero and α 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 routine may also be called with $x = 0$, since $I_q(0) = 0$ when $q > 0$. For negative orders the formula

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

is used to generate the required sequence.

4 Parameters

1: **x** – double *Input*

On entry: the argument x of the function.

Constraint: **x** $\neq 0.0$ when **nl** < 0.

2: **a** – double *Input*

On entry: the order α of the first member in the required sequence of function values.

Constraint: $0.0 \leq$ **a** < 1.0.

3: **nl** – Integer *Input*

On entry: the value of N.

Constraint: abs(**nl**) ≤ 101.

4: **b**[*dim1*] – Complex *Output*

Note: the dimension, *dim1*, of the array **b** must be at least abs(**nl**)+1.

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** ≥ 1 and $I_{\alpha-n+1}(x)$ otherwise, for $n = 1, 2, \ldots$,abs(**nl**)+1.

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

The NAG error parameter (see the Essential Introduction).

5 Error Indicators and Warnings

NE_REAL_INT

On entry, **x** = <*value*>, **nl** = <*value*>.
Constraint: **x** $\neq$ 0.0 when **nl** $<$ 0.

NE_REAL

On entry, **a** = <*value*>.
Constraint: 0.0 $\leq$ **a** $<$ 1.0.

NE_INT

On entry, **nl** = <*value*>.
Constraint: abs(**nl**) $\leq$ 101.

NE_OVERFLOW_LIKELY

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

NW_SOME_PRECISION_LOSS

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

NE_TOTAL_PRECISION_LOSS

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

NE_TERMINATION_FAILURE

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

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 consult NAG for assistance.

6 Further Comments

6.1 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.

6.2 References

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

7 See Also

None.

8 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.

8.1 Program Text

```
/* nag_bessel_i_alpha (s18ejc) Example Program.
 *
 * Copyright 2000 Numerical Algorithms Group.
 *
 * NAG C Library
 *
 * Mark 6, 2000.
 */

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

static double c_1 = 1.;

int main(void)
{
  Complex b[101];
  double a;
  double alpha;
  double d__1;
  double x;
  Integer i;
  Integer exit_status=0;
  Integer nl;
  NagError fail;

  INIT_FAIL(fail);
  Vprintf("s18ejc Example Program Results\n\n");
  /*  Skip heading in data file */
  Vscanf("%*[^\n]");
  while (scanf("%lf %lf %ld%*[^\n]", &x, &a, &nl) != EOF)
    {
      Vprintf("\n  x     a      nl\n\n");
      Vprintf("%4.1f  %4.1f %6ld\n\n", x, a, nl);
      s18ejc  (x, a, nl, b, &fail);
      if (fail.code == NE_NOERROR)
        {
          Vprintf("\n Requested values of I_alpha(X)\n\n");
          alpha = a;
          Vprintf("     alpha              I_alpha(X)\n");
          for (i = 1; i <= ABS(nl) + 1; ++i)
            {
              Vprintf("%12.4e   (%12.4e, %12.4e)\n",
                      alpha, b[i - 1].re, b[i - 1].im);
              d__1 = (double) nl;
              alpha += SIGN (c_1, d__1);
            }
        }
      else
        {
          Vprintf("Error from s18ejc.\n%s\n", fail.message);
          exit_status = 1;
          goto END;
        }
    }
```

```
 END:
  return exit_status;
}    /*  main */
```

8.2 Program Data

```
s18ejc Example Program Data
 0.5   0.0   3 : Values of x, a and nl
```

8.3 Program Results

```
s18ejc Example Program Results


  x     a      nl

 0.5   0.0      3


 Requested values of I_alpha(X)

     alpha              I_alpha(X)
  0.0000e+00   (  1.0635e+00,   0.0000e+00)
  1.0000e+00   (  2.5789e-01,   0.0000e+00)
  2.0000e+00   (  3.1906e-02,   0.0000e+00)
  3.0000e+00   (  2.6451e-03,   0.0000e+00)
```