NAG Library Function Document

nag_bessel_j_alpha (s18ekc)

1 Purpose

nag_bessel_j_alpha (s18ekc) returns a sequence of values for the Bessel functions $J_{\alpha+n-1}(x)$ or $J_{\alpha-n+1}(x)$ for real x, non-negative $\alpha < 1$ and $n = 1, 2, \ldots, |N| + 1$.

2 Specification

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

3 Description

nag_bessel_j_alpha (s18ekc) evaluates a sequence of values for the Bessel function of the first kind $J_\alpha(x)$, where x is real and nonzero and α is the order with $0 \le \alpha < 1$. The $(|N| + 1)$-member sequence is generated for orders $\alpha, \alpha + 1, \ldots, \alpha + N$when $N \ge 0$. Note that $+$ is replaced by $-$ when $N < 0$. For positive orders the function may also be called with $x = 0$, since $J_q(0) = 0$ when $q > 0$. For negative orders the formula

$$J_{-q}(x) = \cos(\pi q) J_q(x) - \sin(\pi q) Y_q(x)$$

is used to generate the required sequence.

4 References

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

5 Arguments

1: **x** – double *Input*

On entry: the argument x of the function.

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

2: **a** – double *Input*

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

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

3: **nl** – Integer *Input*

On entry: the value of N.

Constraint: abs(**nl**) ≤ 101.

4: **b**[$\times$] – Complex *Output*

On exit: with **fail.code** = NE_NOERROR or **fail.code** = NW_SOME_PRECISION_LOSS, the required sequence of function values: **b**(n) contains $J_{\alpha+n-1}(x)$ if **nl** ≥ 0 and $J_{\alpha-n+1}(x)$ otherwise, for $n = 1, 2, \ldots,$ abs(**nl**) $+ 1$.

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

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

6 Error Indicators and Warnings

NE_INT

On entry, **nl** = ⟨*value*⟩.
Constraint: abs(**nl**) ≤ 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** = ⟨*value*⟩.
Constraint: $0.0 \leq$ **a** < 1.0.

NE_REAL_INT

On entry, **x** = ⟨*value*⟩, **nl** = ⟨*value*⟩.
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 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 are, 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 $J_0(x), J_1(x), J_2(x)$ and $J_3(x)$ at $x = 0.5$, and prints the results.

10.1 Program Text

```
/* nag_bessel_j_alpha (s18ekc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * NAG C Library
 *
 * Mark 6, 2000.
 * Mark 8 revised, 2004.
 *
 */

#include <stdio.h>
#include <nag.h>
#include <nag_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_j_alpha (s18ekc) Example Program Results\n");
  if (!(b = NAG_ALLOC(101, Complex)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }
#ifdef _WIN32
  while (scanf_s("%lf %lf %"NAG_IFMT"%*[^\n]", &x, &a, &nl) != EOF)
#else
  while (scanf("%lf %lf %"NAG_IFMT"%*[^\n]", &x, &a, &nl) != EOF)
#endif
    {
      printf("  x     a      nl\n");
      printf("%4.1f  %4.1f %6"NAG_IFMT"\n\n", x, a, nl);
      /* nag_bessel_j_alpha (s18ekc).
       * Bessel functions J_(alpha + n - 1)(x) or
       * J_(alpha - n + 1)(x) for real x != 0, non-negative
       * alpha < 1 and n = 1 , 2 , ... , |N| + 1
       */
      nag_bessel_j_alpha(x, a, nl, b, &fail);
      if (fail.code == NE_NOERROR)
        {
          printf(" Requested values of J_alpha(X)\n\n");
          alpha = a;
          printf("     alpha              J_alpha(X)\n");
          for (i = 0; i < ABS(nl) + 1; ++i)
            {
              printf(" %13.4e   (%13.4e, %13.4e)\n", alpha,
                      b[i].re, b[i].im);
              d = (double) nl;
              alpha += SIGN(1.0, d);
            }
        }
      else
        {
          printf("Error from nag_bessel_j_alpha (s18ekc).\n%s\n",
                  fail.message);
          exit_status = 1;
          goto END;
        }
    }
 END:
  NAG_FREE(b);
  return exit_status;
}
```

10.2 Program Data

```
nag_bessel_j_alpha (s18ekc) Example Program Data
 0.5   0.0   3 : Values of x, a and nl
```

10.3 Program Results

```
nag_bessel_j_alpha (s18ekc) Example Program Results
  x     a      nl
 0.5   0.0      3

 Requested values of J_alpha(X)

     alpha              J_alpha(X)
   0.0000e+00   (  9.3847e-01,   0.0000e+00)
   1.0000e+00   (  2.4227e-01,   0.0000e+00)
   2.0000e+00   (  3.0604e-02,   0.0000e+00)
   3.0000e+00   (  2.5637e-03,   0.0000e+00)
```