

# NAG Library Function Document

## nag\_bessel\_k\_nu (s18efc)

### 1 Purpose

nag\_bessel\_k\_nu (s18efc) returns the value of the modified Bessel function  $K_{\nu/4}(x)$  for real  $x > 0$ .

### 2 Specification

```
#include <nag.h>
#include <nags.h>
double nag_bessel_k_nu (double x, Integer nu, NagError *fail)
```

### 3 Description

nag\_bessel\_k\_nu (s18efc) evaluates an approximation to the modified Bessel function of the second kind  $K_{\nu/4}(x)$ , where the order  $\nu = -3, -2, -1, 1, 2$  or  $3$  and  $x$  is real and positive. For negative orders the formula

$$K_{-\nu/4}(x) = K_{\nu/4}(x)$$

is used.

### 4 References

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

### 5 Arguments

- |    |   |                     |
|----|---|---------------------|
| 1: | <b>x</b> – double   | <i>Input</i>        |
|    | <i>On entry:</i> the argument $x$ of the function.  |                     |
|    | <i>Constraint:</i> $x > 0.0$ .  |                     |
| 2: | <b>nu</b> – Integer   | <i>Input</i>        |
|    | <i>On entry:</i> the argument $\nu$ of the function.  |                     |
|    | <i>Constraint:</i> $1 \leq \text{abs}(\mathbf{nu}) \leq 3$ .                                  |                     |
| 3: | <b>fail</b> – NagError *  | <i>Input/Output</i> |
|    | The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation). |                     |

### 6 Error Indicators and Warnings

#### NE\_INT

On entry, **nu** =  $\langle value \rangle$ .  
 Constraint:  $1 \leq \text{abs}(\mathbf{nu}) \leq 3$ .

**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. The result is returned as zero.

**NE\_REAL**

On entry,  $x = \langle value \rangle$ .  
Constraint:  $x > 0.0$ .

**NE\_TERMINATION\_FAILURE**

The evaluation has been abandoned due to failure to satisfy the termination condition. The result is returned as zero.

**NE\_TOTAL\_PRECISION\_LOSS**

The evaluation has been abandoned due to total loss of precision. The result is returned as zero.

**NW\_SOME\_PRECISION\_LOSS**

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

**7 Accuracy**

All constants in the underlying function 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 function, the actual number of correct digits is limited, in general, by  $p - s$ , where  $s \approx \max(1, |\log_{10} x|)$  represents the number of digits lost due to the argument reduction. Thus the larger the value of  $x$ , the less the precision in the result.

**8 Parallelism and Performance**

nag\_bessel\_k\_nu (s18efc) is not threaded in any implementation.

**9 Further Comments**

None.

**10 Example**

The example program reads values of the arguments  $x$  and  $\nu$  from a file, evaluates the function and prints the results.

**10.1 Program Text**

```
/* nag_bessel_k_nu (s18efc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * NAG C Library
 *
 * Mark 26, 2016.
```

```

*/

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

int main(void)
{
    Integer exit_status = 0, nu;
    NagError fail;
    double x, y;

    INIT_FAIL(fail);

    /* Skip heading in data file */
#ifdef _WIN32
    scanf_s("%*[\n]");
#else
    scanf("%*[\n]");
#endif
    printf("nag_bessel_k_nu (s18efc) Example Program Results\n");
    printf("      x          nu          y\n");
#ifdef _WIN32
    while (scanf_s("%lf %" NAG_IFMT "%*[\n]", &x, &nu) != EOF)
#else
    while (scanf("%lf %" NAG_IFMT "%*[\n]", &x, &nu) != EOF)
#endif
    {
        /* nag_bessel_k_nu (s18efc).
         * Modified Bessel function Kν(x)
         */
        y = nag_bessel_k_nu(x, nu, &fail);
        if (fail.code != NE_NOERROR) {
            printf("Error from nag_bessel_k_nu (s18efc).\n%s\n", fail.message);
            exit_status = 1;
            goto END;
        }
        printf("%4.1f %6" NAG_IFMT " %13.4e\n", x, nu, y);
    }
END:
    return exit_status;
}

```

## 10.2 Program Data

nag\_bessel\_k\_nu (s18efc) Example Program Data

```

3.9  -3
1.4  -2
8.2  -1
6.7   1
0.5   2
2.3   3 : Values of x and nu

```

## 10.3 Program Results

nag\_bessel\_k\_nu (s18efc) Example Program Results

```

      x          nu          y
3.9    -3      1.3315e-02
1.4    -2      2.6121e-01
8.2    -1      1.1892e-04
6.7     1      5.8826e-04
0.5     2      1.0750e+00
2.3     3      8.7724e-02

```

---