

# NAG Library Function Document

## nag\_bessel\_k0 (s18acc)

### 1 Purpose

nag\_bessel\_k0 (s18acc) returns the value of the modified Bessel function  $K_0(x)$ .

### 2 Specification

```
#include <nag.h>
#include <nags.h>
double nag_bessel_k0 (double x, NagError *fail)
```

### 3 Description

nag\_bessel\_k0 (s18acc) evaluates an approximation to the modified Bessel function of the second kind  $K_0(x)$ .

**Note:**  $K_0(x)$  is undefined for  $x \leq 0$  and the function will fail for such arguments.

The function is based on five Chebyshev expansions:

For  $0 < x \leq 1$ ,

$$K_0(x) = -\ln x \sum_{r=0} a_r T_r(t) + \sum_{r=0} b_r T_r(t), \quad \text{where } t = 2x^2 - 1.$$

For  $1 < x \leq 2$ ,

$$K_0(x) = e^{-x} \sum_{r=0} c_r T_r(t), \quad \text{where } t = 2x - 3.$$

For  $2 < x \leq 4$ ,

$$K_0(x) = e^{-x} \sum_{r=0} d_r T_r(t), \quad \text{where } t = x - 3.$$

For  $x > 4$ ,

$$K_0(x) = \frac{e^{-x}}{\sqrt{x}} \sum_{r=0} e_r T_r(t), \quad \text{where } t = \frac{9-x}{1+x}.$$

For  $x$  near zero,  $K_0(x) \simeq -\gamma - \ln\left(\frac{x}{2}\right)$ , where  $\gamma$  denotes Euler's constant. This approximation is used when  $x$  is sufficiently small for the result to be correct to *machine precision*.

For large  $x$ , where there is a danger of underflow due to the smallness of  $K_0$ , the result is set exactly to zero.

### 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:*  $x > 0.0$ .
- 2: **fail** – NagError \* *Input/Output*  
 The NAG error argument (see Section 3.6 in the Essential Introduction).

## 6 Error Indicators and Warnings

### NE\_ALLOC\_FAIL

Dynamic memory allocation failed.  
 See Section 3.2.1.2 in the Essential Introduction for further information.

### 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.

### NE\_REAL\_ARG\_LE

*On entry,*  $x = \langle value \rangle$ .  
*Constraint:*  $x > 0.0$ .  
 $K_0$  is undefined and the function returns zero.

## 7 Accuracy

Let  $\delta$  and  $\epsilon$  be the relative errors in the argument and result respectively.

If  $\delta$  is somewhat larger than the *machine precision* (i.e., if  $\delta$  is due to data errors etc.), then  $\epsilon$  and  $\delta$  are approximately related by:

$$\epsilon \simeq \left| \frac{xK_1(x)}{K_0(x)} \right| \delta.$$

Figure 1 shows the behaviour of the error amplification factor

$$\left| \frac{xK_1(x)}{K_0(x)} \right|.$$

However, if  $\delta$  is of the same order as *machine precision*, then rounding errors could make  $\epsilon$  slightly larger than the above relation predicts.

For small  $x$ , the amplification factor is approximately  $\left| \frac{1}{\ln x} \right|$ , which implies strong attenuation of the error, but in general  $\epsilon$  can never be less than the *machine precision*.

For large  $x$ ,  $\epsilon \simeq x\delta$  and we have strong amplification of the relative error. Eventually  $K_0$ , which is asymptotically given by  $\frac{e^{-x}}{\sqrt{x}}$ , becomes so small that it cannot be calculated without underflow and hence

the function will return zero. Note that for large  $x$  the errors will be dominated by those of the standard math library function `exp`.

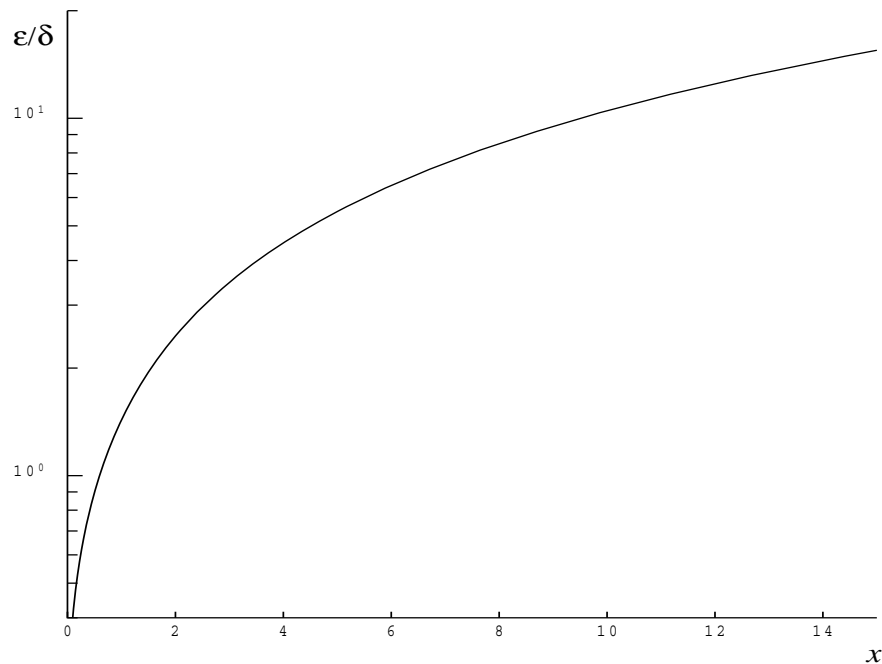


Figure 1

## 8 Parallelism and Performance

Not applicable.

## 9 Further Comments

None.

## 10 Example

This example reads values of the argument  $x$  from a file, evaluates the function at each value of  $x$  and prints the results.

### 10.1 Program Text

```
/* nag_bessel_k0 (s18acc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 2 revised, 1992.
 */

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

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

    INIT_FAIL(fail);
```

```

/* Skip heading in data file */
#ifdef _WIN32
scanf_s("%*[\n]");
#else
scanf("%*[\n]");
#endif
printf("nag_bessel_k0 (s18acc) Example Program Results\n");
printf("      x          y\n");
#ifdef _WIN32
while (scanf_s("%lf", &x) != EOF)
#else
while (scanf("%lf", &x) != EOF)
#endif
{
/* nag_bessel_k0 (s18acc).
* Modified Bessel function K_0(x)
*/
y = nag_bessel_k0(x, &fail);
if (fail.code != NE_NOERROR)
{
printf("Error from nag_bessel_k0 (s18acc).\n%s\n",
fail.message);
exit_status = 1;
goto END;
}
printf("%12.3e%12.3e\n", x, y);
}

END:
return exit_status;
}

```

## 10.2 Program Data

```

nag_bessel_k0 (s18acc) Example Program Data
0.4
0.6
1.4
1.6
2.5
3.5
6.0
8.0
10.0
1000.0

```

## 10.3 Program Results

```

nag_bessel_k0 (s18acc) Example Program Results
      x          y
4.000e-01  1.115e+00
6.000e-01  7.775e-01
1.400e+00  2.437e-01
1.600e+00  1.880e-01
2.500e+00  6.235e-02
3.500e+00  1.960e-02
6.000e+00  1.244e-03
8.000e+00  1.465e-04
1.000e+01  1.778e-05
1.000e+03  0.000e+00

```

**Example Program**  
Returned Values for the Bessel Function  $K_0(x)$

