NAG Library Function Document
nag_bessel_k1 (s18adc)

1 Purpose
nag_bessel_k1 (s18adc) returns the value of the modified Bessel function \( K_1(x) \).

2 Specification
#include <nag.h>
#include <nags.h>
double nag_bessel_k1 (double x, NagError *fail)

3 Description
nag_bessel_k1 (s18adc) evaluates an approximation to the modified Bessel function of the second kind \( K_1(x) \).

Note: \( K_1(x) \) is undefined for \( x \leq 0 \) and the function will fail for such arguments.

The function is based on five Chebyshev expansions:
For \( 0 < x \leq 1 \),
\[
K_1(x) = \frac{1}{x} + x \ln x \sum_{r=0}^\infty a_r T_r(t) - x \sum_{r=0}^\infty b_r T_r(t), \quad \text{where } t = 2x^2 - 1.
\]
For \( 1 < x \leq 2 \),
\[
K_1(x) = e^{-x} \sum_{r=0}^\infty c_r T_r(t), \quad \text{where } t = 2x - 3.
\]
For \( 2 < x \leq 4 \),
\[
K_1(x) = e^{-x} \sum_{r=0}^\infty d_r T_r(t), \quad \text{where } t = x - 3.
\]
For \( x > 4 \),
\[
K_1(x) = e^{-x} \sqrt{x} \sum_{r=0}^\infty e_r T_r(t), \quad \text{where } t = \frac{9 - x}{1 + x}.
\]
For \( x \) near zero, \( K_1(x) \approx \frac{1}{x} \). This approximation is used when \( x \) is sufficiently small for the result to be correct to machine precision. For very small \( x \) on some machines, it is impossible to calculate \( \frac{1}{x} \) without overflow and the function must fail.
For large \( x \), where there is a danger of underflow due to the smallness of \( K_1 \), the result is set exactly to zero.

4 References
5 Arguments

1:  \( x \) – double

\textit{Input}

\textit{On entry}: the argument \( x \) of the function.

\textit{Constraint}: \( x > 0.0 \).

2:  \textit{fail} – NagError \(*\)

\textit{Input/Output}

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

6 Error Indicators and Warnings

\textbf{NE_ALLOC_FAIL}

Dynamic memory allocation failed.

See Section 3.2.1.2 in the Essential Introduction for further information.

\textbf{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.

\textbf{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.

\textbf{NE_REAL_ARG_LE}

On entry, \( x = \langle \text{value} \rangle \).

\textit{Constraint}: \( x > 0.0 \).

\( K_0 \) is undefined and the function returns zero.

\textbf{NE_REAL_ARG_TOO_SMALL}

On entry, \( x = \langle \text{value} \rangle \).

\textit{Constraint}: \( x > \langle \text{value} \rangle \).

\( x \) is too small, there is a danger of overflow and the function returns approximately the largest representable value.

7 Accuracy

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

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

\[
\epsilon \simeq \left| \frac{xK_0(x) - K_1(x)}{K_1(x)} \right| \delta.
\]

Figure 1 shows the behaviour of the error amplification factor

\[
\left| \frac{xK_0(x) - K_1(x)}{K_1(x)} \right|.
\]

However if \( \delta \) is of the same order as the \textit{machine precision}, then rounding errors could make \( \epsilon \) slightly larger than the above relation predicts.

For small \( x \), \( \epsilon \simeq \delta \) and there is no amplification of errors.
For large $x$, $e \simeq xb$ and we have strong amplification of the relative error. Eventually $K_1$, which is asymptotically given by $e^{-x}/\sqrt{x}$, 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 function $\exp$.

![Figure 1](image)

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

```c
/* nag_bessel_k1 (s18adc) 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;
    // Some code here
}
INIT_FAIL(fail);

/* Skip heading in data file */
#ifdef _WIN32
    scanf_s("%*[\n]");
#else
    scanf("%*[\n]");
#endif
printf("nag_bessel_k1 (s18adc) 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_k1 (s18adc).
* Modified Bessel function K_1(x)
*/
y = nag_bessel_k1(x, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_bessel_k1 (s18adc).\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_k1 (s18adc) 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_k1 (s18adc) Example Program Results

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.000e-01</td>
<td>2.184e+00</td>
</tr>
<tr>
<td>6.000e-01</td>
<td>1.303e+00</td>
</tr>
<tr>
<td>1.400e+00</td>
<td>3.208e-01</td>
</tr>
<tr>
<td>1.600e+00</td>
<td>2.406e-01</td>
</tr>
<tr>
<td>2.500e+00</td>
<td>7.389e-02</td>
</tr>
<tr>
<td>3.500e+00</td>
<td>2.224e-02</td>
</tr>
<tr>
<td>6.000e+00</td>
<td>1.344e-03</td>
</tr>
<tr>
<td>8.000e+00</td>
<td>1.554e-04</td>
</tr>
<tr>
<td>1.000e+01</td>
<td>1.865e-05</td>
</tr>
<tr>
<td>1.000e+03</td>
<td>0.000e+00</td>
</tr>
</tbody>
</table>
Example Program
Returned Values for the Bessel Function $K_1(x)$