NAG Library Function Document

nag_bessel_k1_vector (s18arc)

1 Purpose

nag_bessel_k1_vector (s18arc) returns an array of values of the modified Bessel function $K_1(x)$.

2 Specification

```
#include <nag.h>
#include <nags.h>
void nag_bessel_k1_vector (Integer n, const double x[], double f[],
     Integer ivalid[], NagError *fail)
```

3 Description

nag_bessel_k1_vector (s18arc) evaluates an approximation to the modified Bessel function of the second kind $K_1(x_i)$ for an array of arguments x_i, for $i = 1, 2, \dots, n$.

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 $0 < x \leq 1$,

$$K_1(x) = \frac{1}{x} + x \ln x \sum_{r=0} a_r T_r(t) - x \sum_{r=0} b_r T_r(t), \qquad \text{where } t = 2x^2 - 1.$$

For $1 < x \leq 2$,

$$K_1(x) = e^{-x} \sum_{r=0} c_r T_r(t), \qquad \text{where } t = 2x - 3.$$

For $2 < x \leq 4$,

$$K_1(x) = e^{-x} \sum_{r=0} d_r T_r(t), \qquad \text{where } t = x - 3.$$

For $x > 4$,

$$K_1(x) = \frac{e^{-x}}{\sqrt{x}} \sum_{r=0} e_r T_r(t), \qquad \text{where } t = \frac{9 - x}{1 + x}.$$

For x near zero, $K_1(x) \simeq \frac{1}{x}$. This approximation is used when x is sufficiently small for the result to be correct to ***machine precision***. For very small x 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

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

5 Arguments

1: **n** – Integer *Input*

On entry: n, the number of points.

Constraint: **n** ≥ 0.

2: **x**[**n**] – const double *Input*

On entry: the argument x_i of the function, for $i = 1, 2, \ldots,$ **n**.

Constraint: **x**$[i - 1] > 0.0$, for $i = 1, 2, \ldots,$ **n**.

3: **f**[**n**] – double *Output*

On exit: $K_1(x_i)$, the function values.

4: **ivalid**[**n**] – Integer *Output*

On exit: **ivalid**$[i - 1]$ contains the error code for x_i, for $i = 1, 2, \ldots,$ **n**.

ivalid$[i - 1] = 0$
No error.

ivalid$[i - 1] = 1$
$x_i \leq 0.0$, $K_1(x_i)$ is undefined. **f**$[i - 1]$ contains 0.0.

ivalid$[i - 1] = 2$
x_i is too small, there is a danger of overflow. **f**$[i - 1]$ contains zero. The threshold value is the same as for **fail.code** = NE_REAL_ARG_TOO_SMALL in nag_bessel_k1 (s18adc), as defined in the Users' Note for your implementation.

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

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

6 Error Indicators and Warnings

NE_BAD_PARAM

On entry, argument ⟨*value*⟩ had an illegal value.

NE_INT

On entry, **n** = ⟨*value*⟩.
Constraint: **n** ≥ 0.

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.

NW_IVALID

On entry, at least one value of **x** was invalid.
Check **ivalid** for more information.

7 Accuracy

Let δ and ϵ be the relative errors in the argument and result respectively.

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

$$\epsilon \simeq \left| \frac{x K_0(x) - K_1(x)}{K_1(x)} \right| \delta.$$

Figure 1 shows the behaviour of the error amplification factor

$$\left| \frac{x K_0(x) - K_1(x)}{K_1(x)} \right|.$$

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

For small x, $\epsilon \simeq \delta$ and there is no amplification of errors.

For large x, $\epsilon \simeq x\delta$ and we have strong amplification of the relative error. Eventually K_1, 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 function exp.

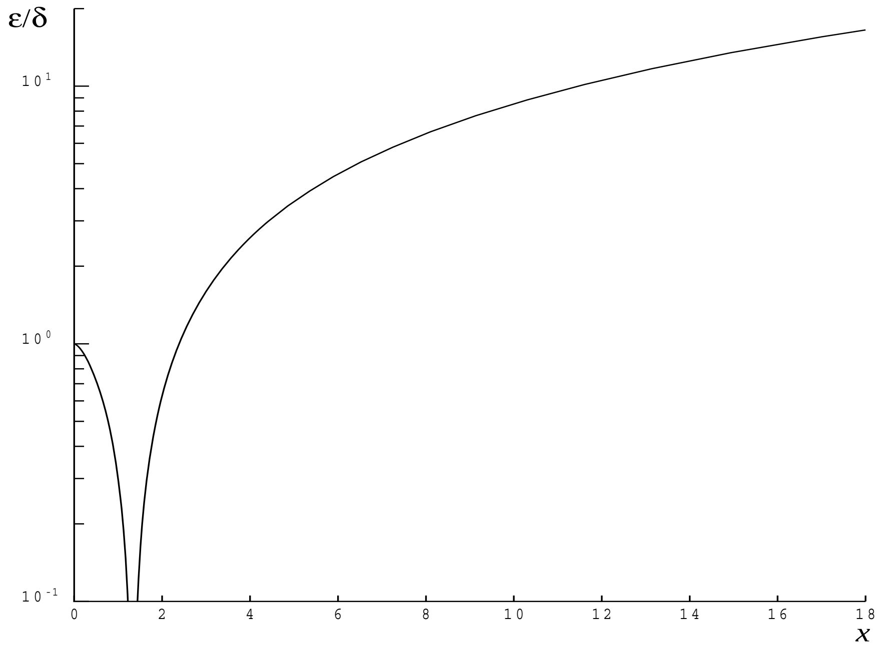

Figure 1

8 Parallelism and Performance

Not applicable.

9 Further Comments

None.

10 Example

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

10.1 Program Text

```
/* nag_bessel_k1_vector (s18arc) Example Program.
 *
 * Copyright 2011, Numerical Algorithms Group.
 *
 * Mark 23 2011.
 */
#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nags.h>

int main(void)
{
  Integer  exit_status = 0;
  Integer  i, n;
  double   *f = 0, *x = 0;
  Integer  *ivalid = 0;
  NagError fail;

  INIT_FAIL(fail);

  /* Skip heading in data file */
  scanf("%*[^\n]");

  printf("nag_bessel_k1_vector (s18arc) Example Program Results\n");
  printf("\n");
  printf("     x           f          ivalid\n");
  printf("\n");
  scanf("%ld", &n);
  scanf("%*[^\n]");

  /* Allocate memory */
  if (!(x = NAG_ALLOC(n, double)) ||
      !(f = NAG_ALLOC(n, double)) ||
      !(ivalid = NAG_ALLOC(n, Integer)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  for (i=0; i<n; i++)
    scanf("%lf", &x[i]);
  scanf("%*[^\n]");

  /* nag_bessel_k1_vector (s18arc).
   * modified Bessel Function K1(x)
   */
  nag_bessel_k1_vector(n, x, f, ivalid, &fail);
  if (fail.code!=NE_NOERROR && fail.code!=NW_IVALID)
    {
      printf("Error from nag_bessel_k1_vector (s18arc).\n%s\n",
             fail.message);
      exit_status = 1;
      goto END;
    }

  for (i=0; i<n; i++)
    printf(" %11.3e %11.3e %4ld\n", x[i], f[i], ivalid[i]);

 END:
  NAG_FREE(f);
  NAG_FREE(x);
  NAG_FREE(ivalid);

  return exit_status;
}
```

10.2 Program Data

```
nag_bessel_k1_vector (s18arc) Example Program Data

10

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_vector (s18arc) Example Program Results

     x           f          ivalid

   4.000e-01   2.184e+00    0
   6.000e-01   1.303e+00    0
   1.400e+00   3.208e-01    0
   1.600e+00   2.406e-01    0
   2.500e+00   7.389e-02    0
   3.500e+00   2.224e-02    0
   6.000e+00   1.344e-03    0
   8.000e+00   1.554e-04    0
   1.000e+01   1.865e-05    0
   1.000e+03   0.000e+00    0
```