

## NAG Library Function Document

### nag\_kelvin\_ber\_vector (s19anc)

#### 1 Purpose

nag\_kelvin\_ber\_vector (s19anc) returns an array of values for the Kelvin function ber  $x$ .

#### 2 Specification

```
#include <nag.h>
#include <nags.h>
```

```
void nag_kelvin_ber_vector (Integer n, const double x[], double f[],
    Integer ivalid[], NagError *fail)
```

#### 3 Description

nag\_kelvin\_ber\_vector (s19anc) evaluates an approximation to the Kelvin function ber  $x_i$  for an array of arguments  $x_i$ , for  $i = 1, 2, \dots, n$ .

**Note:** ber( $-x$ ) = ber  $x$ , so the approximation need only consider  $x \geq 0.0$ .

The function is based on several Chebyshev expansions:

For  $0 \leq x \leq 5$ ,

$$\text{ber } x = \sum_{r=0} a_r T_r(t), \quad \text{with } t = 2\left(\frac{x}{5}\right)^4 - 1.$$

For  $x > 5$ ,

$$\begin{aligned} \text{ber } x = & \frac{e^{x/\sqrt{2}}}{\sqrt{2\pi x}} \left[ \left(1 + \frac{1}{x}a(t)\right) \cos \alpha + \frac{1}{x}b(t) \sin \alpha \right] \\ & + \frac{e^{-x/\sqrt{2}}}{\sqrt{2\pi x}} \left[ \left(1 + \frac{1}{x}c(t)\right) \sin \beta + \frac{1}{x}d(t) \cos \beta \right], \end{aligned}$$

where  $\alpha = \frac{x}{\sqrt{2}} - \frac{\pi}{8}$ ,  $\beta = \frac{x}{\sqrt{2}} + \frac{\pi}{8}$ ,

and  $a(t)$ ,  $b(t)$ ,  $c(t)$ , and  $d(t)$  are expansions in the variable  $t = \frac{10}{x} - 1$ .

When  $x$  is sufficiently close to zero, the result is set directly to ber 0 = 1.0.

For large  $x$ , there is a danger of the result being totally inaccurate, as the error amplification factor grows in an essentially exponential manner; therefore the function must fail.

#### 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:*  $n \geq 0$ .

- 2: **x[n]** – const double *Input*  
*On entry:* the argument  $x_i$  of the function, for  $i = 1, 2, \dots, \mathbf{n}$ .
- 3: **f[n]** – double *Output*  
*On exit:* ber  $x_i$ , the function values.
- 4: **ivalid[n]** – Integer *Output*  
*On exit:* **ivalid**[ $i - 1$ ] contains the error code for  $x_i$ , for  $i = 1, 2, \dots, \mathbf{n}$ .  
**ivalid**[ $i - 1$ ] = 0  
 No error.  
**ivalid**[ $i - 1$ ] = 1  
 $\text{abs}(x_i)$  is too large for an accurate result to be returned. **f**[ $i - 1$ ] contains zero. The threshold value is the same as for **fail.code** = NE\_REAL\_ARG\_GT in nag\_kelvin\_ber (s19aac), as defined in the Users' Note for your implementation.
- 5: **fail** – NagError \* *Input/Output*  
 The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

## 6 Error Indicators and Warnings

### NE\_ALLOC\_FAIL

Dynamic memory allocation failed.

See Section 2.3.1.2 in How to Use the NAG Library and its Documentation for further information.

### NE\_BAD\_PARAM

On entry, argument  $\langle \text{value} \rangle$  had an illegal value.

### NE\_INT

On entry,  $\mathbf{n} = \langle \text{value} \rangle$ .

Constraint:  $\mathbf{n} \geq 0$ .

### 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.

An unexpected error has been triggered by this function. Please contact NAG.

See Section 2.7.6 in How to Use the NAG Library and its Documentation for further information.

### NE\_NO\_LICENCE

Your licence key may have expired or may not have been installed correctly.

See Section 2.7.5 in How to Use the NAG Library and its Documentation for further information.

### NW\_INVALID

On entry, at least one value of **x** was invalid.

Check **ivalid** for more information.

## 7 Accuracy

Since the function is oscillatory, the absolute error rather than the relative error is important. Let  $E$  be the absolute error in the result and  $\delta$  be the relative error in the argument. If  $\delta$  is somewhat larger than the *machine precision*, then we have:

$$E \simeq \left| \frac{x}{\sqrt{2}} (\text{ber}_1 x + \text{bei}_1 x) \right| \delta$$

(provided  $E$  is within machine bounds).

For small  $x$  the error amplification is insignificant and thus the absolute error is effectively bounded by the *machine precision*.

For medium and large  $x$ , the error behaviour is oscillatory and its amplitude grows like  $\sqrt{\frac{x}{2\pi}} e^{x/\sqrt{2}}$ .

Therefore it is not possible to calculate the function with any accuracy when  $\sqrt{x} e^{x/\sqrt{2}} > \frac{\sqrt{2\pi}}{\delta}$ . Note that this value of  $x$  is much smaller than the minimum value of  $x$  for which the function overflows.

## 8 Parallelism and Performance

nag\_kelvin\_ber\_vector (s19anc) is not threaded in any implementation.

## 9 Further Comments

None.

## 10 Example

This example reads values of  $\mathbf{x}$  from a file, evaluates the function at each value of  $x_i$  and prints the results.

### 10.1 Program Text

```

/* nag_kelvin_ber_vector (s19anc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */
#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 */
#ifdef _WIN32
    scanf_s("%*[^\\n]");
#else
    scanf("%*[^\\n]");
#endif
}

```

```

    printf("nag_kelvin_ber_vector (s19anc) Example Program Results\n");
    printf("\n");
    printf("      x          f          ivalid\n");
    printf("\n");
#ifdef _WIN32
    scanf_s("%" NAG_IFMT " ", &n);
#else
    scanf("%" NAG_IFMT " ", &n);
#endif
#ifdef _WIN32
    scanf_s("%*[\n]");
#else
    scanf("%*[\n]");
#endif

    /* 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++)
#ifdef _WIN32
        scanf_s("%lf", &x[i]);
#else
        scanf("%lf", &x[i]);
#endif
#ifdef _WIN32
    scanf_s("%*[\n]");
#else
    scanf("%*[\n]");
#endif

    /* nag_kelvin_ber_vector (s19anc).
     * Kelvin Function ber x
     */
    nag_kelvin_ber_vector(n, x, f, ivalid, &fail);
    if (fail.code != NE_NOERROR && fail.code != NW_INVALID) {
        printf("Error from nag_kelvin_ber_vector (s19anc).\n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }

    for (i = 0; i < n; i++)
        printf(" %11.3e %11.3e %4" NAG_IFMT "\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\_kelvin\_ber\_vector (s19anc) Example Program Data

7

0.1 1.0 2.5 5.0 10.0 15.0 -1.0

### 10.3 Program Results

nag\_kelvin\_ber\_vector (s19anc) Example Program Results

x	f	ivalid
1.000e-01	1.000e+00	0
1.000e+00	9.844e-01	0
2.500e+00	4.000e-01	0
5.000e+00	-6.230e+00	0
1.000e+01	1.388e+02	0
1.500e+01	-2.967e+03	0
-1.000e+00	9.844e-01	0

---