

NAG Library Function Document

nag_bessel_i_nu_scaled (s18ecc)

1 Purpose

nag_bessel_i_nu_scaled (s18ecc) returns the value of the scaled modified Bessel function $e^{-x}I_{\nu/4}(x)$ for real $x > 0$.

2 Specification

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

```
double nag_bessel_i_nu_scaled (double x, Integer nu, NagError *fail)
```

3 Description

nag_bessel_i_nu_scaled (s18ecc) evaluates an approximation to the scaled modified Bessel function of the first kind $e^{-x}I_{\nu/4}(x)$, where the order $\nu = -3, -2, -1, 1, 2$ or 3 and x is real and positive. For positive orders it may also be called with $x = 0$, since $I_{\nu/4}(0) = 0$ when $\nu > 0$. For negative orders the formula

$$I_{-\nu/4}(x) = I_{\nu/4}(x) + \frac{2}{\pi} \sin\left(\frac{\pi\nu}{4}\right) K_{\nu/4}(x)$$

is used prior to multiplication by the scale factor e^{-x} .

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.
Constraints:
 if **nu** < 0, **x** > 0.0;
 if **nu** > 0, **x** ≥ 0.0.
- 2: **nu** – Integer *Input*
On entry: the argument ν of the function.
Constraint: $1 \leq \text{abs}(\mathbf{nu}) \leq 3$.
- 3: **fail** – NagError * *Input/Output*
 The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

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

On entry, $\mathbf{x} = \langle \text{value} \rangle$, $\mathbf{nu} = \langle \text{value} \rangle$.
 Constraint: $\mathbf{x} > 0.0$ when $\mathbf{nu} < 0$.

On entry, $\mathbf{x} = \langle \text{value} \rangle$, $\mathbf{nu} = \langle \text{value} \rangle$.
 Constraint: $\mathbf{x} \geq 0.0$ when $\mathbf{nu} > 0$.

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 functions 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 functions, 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

Not applicable.

9 Further Comments

None.

10 Example

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

10.1 Program Text

```

/* nag_bessel_i_nu_scaled (s18ecc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * NAG C Library
 *
 * Mark 6, 2000.
 * Mark 7, revised, 2001.
 *
 */

#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_i_nu_scaled (s18ecc) 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_i_nu_scaled (s18ecc).
         * Scaled modified Bessel function exp(-x) I_(nu/4)(x)
         */
        y = nag_bessel_i_nu_scaled(x, nu, &fail);
        if (fail.code != NE_NOERROR)
        {
            printf("Error from nag_bessel_i_nu_scaled (s18ecc).\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_i_nu_scaled (s18ecc) 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_i_nu_scaled (s18ecc) Example Program Results

x	nu	y
3.9	-3	1.9272e-01
1.4	-2	3.5767e-01
8.2	-1	1.4103e-01
6.7	1	1.5649e-01
0.5	2	3.5664e-01
2.3	3	2.3748e-01
