

NAG Library Function Document

nag_bessel_i1_vector (s18atc)

1 Purpose

nag_bessel_i1_vector (s18atc) returns an array of values for the modified Bessel function $I_1(x)$.

2 Specification

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

void nag_bessel_i1_vector (Integer n, const double x[], double f[],
                          Integer ivalid[], NagError *fail)
```

3 Description

nag_bessel_i1_vector (s18atc) evaluates an approximation to the modified Bessel function of the first kind $I_1(x_i)$ for an array of arguments x_i , for $i = 1, 2, \dots, n$.

Note: $I_1(-x) = -I_1(x)$, so the approximation need only consider $x \geq 0$.

The function is based on three Chebyshev expansions:

For $0 < x \leq 4$,

$$I_1(x) = x \sum_{r=0} a_r T_r(t), \quad \text{where } t = 2 \left(\frac{x}{4} \right)^2 - 1;$$

For $4 < x \leq 12$,

$$I_1(x) = e^x \sum_{r=0} b_r T_r(t), \quad \text{where } t = \frac{x-8}{4};$$

For $x > 12$,

$$I_1(x) = \frac{e^x}{\sqrt{x}} \sum_{r=0} c_r T_r(t), \quad \text{where } t = 2 \left(\frac{12}{x} \right) - 1.$$

For small x , $I_1(x) \simeq x$. This approximation is used when x is sufficiently small for the result to be correct to *machine precision*.

For large x , the function must fail because $I_1(x)$ cannot be represented without overflow.

4 References

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

5 Arguments

- | | | |
|----|---|--------------|
| 1: | n – Integer
<i>On entry:</i> n , the number of points.
<i>Constraint:</i> $n \geq 0$. | <i>Input</i> |
| 2: | x[n] – const double
<i>On entry:</i> the argument x_i of the function, for $i = 1, 2, \dots, n$. | <i>Input</i> |

- 3: **f[n]** – double *Output*
On exit: $I_1(x_i)$, the function values.
- 4: **ivalid[n]** – Integer *Output*
On exit: **ivalid**[$i - 1$] contains the error code for x_i , for $i = 1, 2, \dots, \mathbf{n}$.
ivalid[$i - 1$] = 0
 No error.
ivalid[$i - 1$] = 1
 x_i is too large. **f**[$i - 1$] contains the approximate value of $I_1(x_i)$ at the nearest valid argument. The threshold value is the same as for **fail.code** = NE_REAL_ARG_GT in nag_bessel_i1 (s18afc), 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 $\langle value \rangle$ had an illegal value.

NE_INT

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

NW_INVALID

On entry, at least one value of \mathbf{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:

$$\epsilon \simeq \left| \frac{xI_0(x) - I_1(x)}{I_1(x)} \right| \delta.$$

Figure 1 shows the behaviour of the error amplification factor

$$\left| \frac{xI_0(x) - I_1(x)}{I_1(x)} \right|.$$

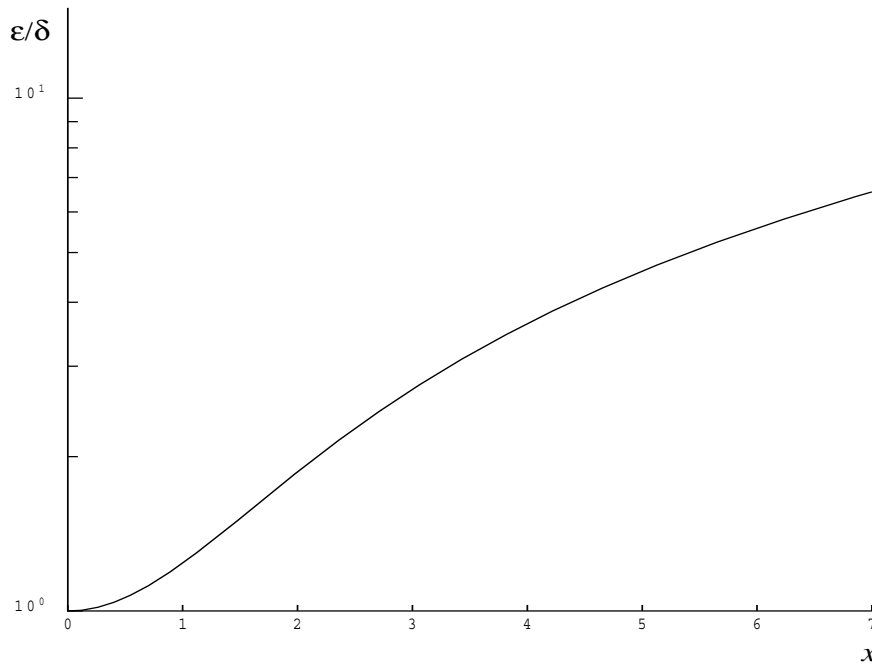


Figure 1

However, if δ is of the same order as *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 errors. However, for quite moderate values of x ($x > \hat{x}$, the threshold value), the function must fail because $I_1(x)$ would overflow; hence in practice the loss of accuracy for x close to \hat{x} is not excessive and the errors will be dominated by those of the standard function \exp .

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_i1_vector (s18atc) 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_il_vector (s18atc) 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_il_vector (s18atc).
 * modified Bessel Function I1(x)
 */
nag_bessel_il_vector(n, x, f, ivalid, &fail);
if (fail.code!=NE_NOERROR && fail.code!=NW_IVALID)
{
    printf("Error from nag_bessel_il_vector (s18atc).\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_il_vector (s18atc) Example Program Data

10

0.0 0.5 1.0 3.0 6.0 8.0 10.0 15.0 20.0 -1.0

10.3 Program Results

nag_bessel_il_vector (s18atc) Example Program Results

x	f	ivalid
0.000e+00	0.000e+00	0
5.000e-01	2.579e-01	0
1.000e+00	5.652e-01	0
3.000e+00	3.953e+00	0
6.000e+00	6.134e+01	0
8.000e+00	3.999e+02	0
1.000e+01	2.671e+03	0
1.500e+01	3.281e+05	0
2.000e+01	4.245e+07	0
-1.000e+00	-5.652e-01	0
