NAG Library Function Document

nag_bessel_i1 (s18afc)

1 Purpose

nag_bessel_i1 (s18afc) returns a value for the modified Bessel function $I_1(x)$.

2 Specification

```
#include <nag.h>
#include <nags.h>
double nag_bessel_i1 (double x, NagError *fail)
```

3 Description

nag_bessel_i1 (s18afc) evaluates an approximation to the modified Bessel function of the first kind $I_1(x)$.

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 $0 < x \leq 4$,

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

For $4 < x \leq 12$,

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

For $x > 12$,

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

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: **x** – double *Input*

On entry: the argument x of the function.

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

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

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

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.

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.

NE_REAL_ARG_GT

On entry, $\mathbf{x} = \langle value \rangle$.
Constraint: $|\mathbf{x}| \leq \langle value \rangle$.
$|\mathbf{x}|$ is too large and the function returns the approximate value of $I_1(x)$ at the nearest valid argument.

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

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

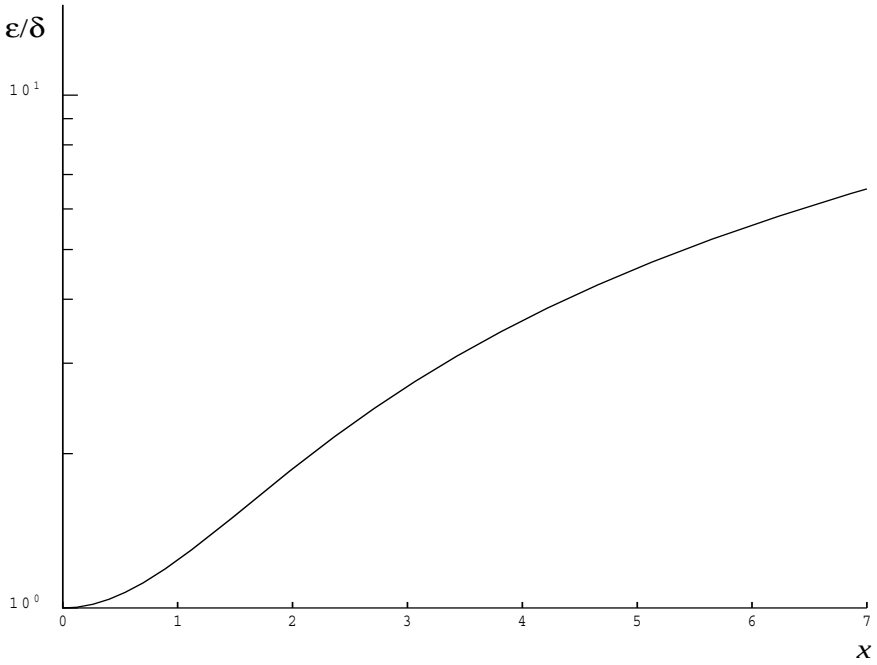


Figure 1

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 the function must fail for quite moderate values of x because $I_1(x)$ would overflow; hence in practice the loss of accuracy for large x is not excessive. Note that for large x, the errors will be dominated by those of the standard math library function exp.

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

```
/* nag_bessel_i1 (s18afc) 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)
```

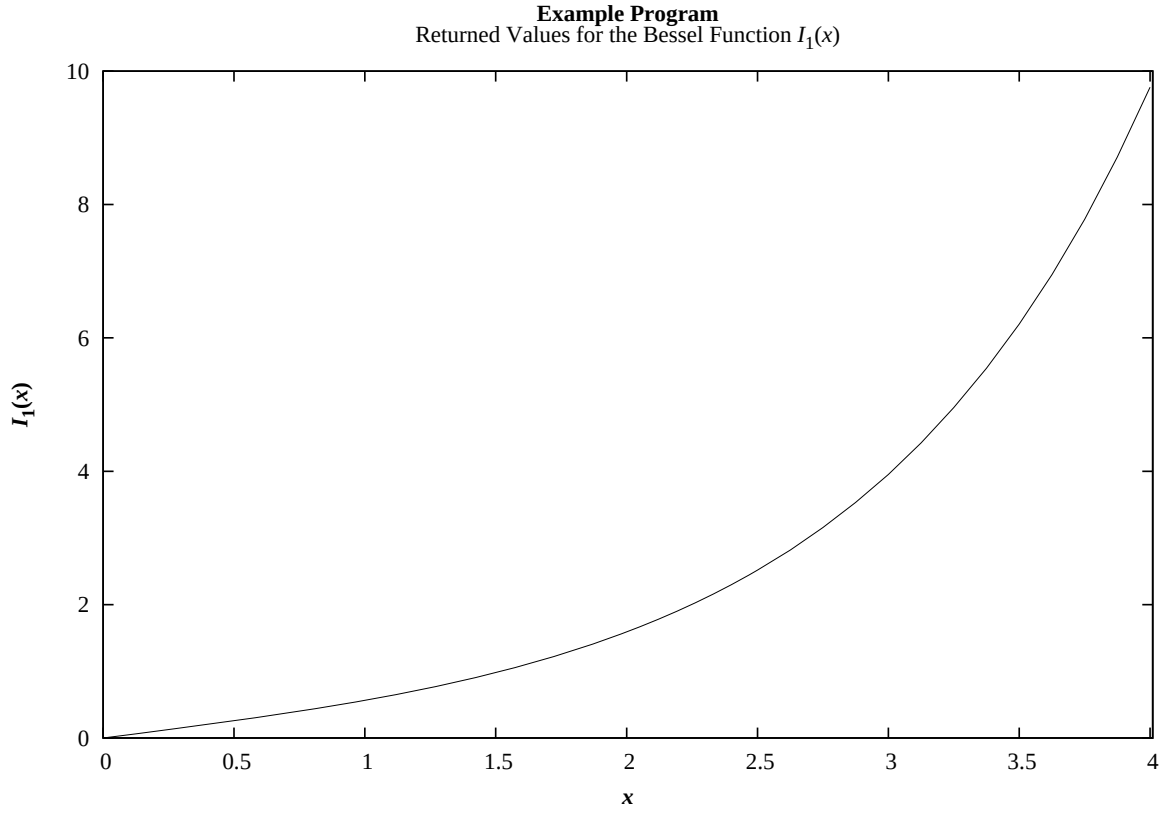
```
{
  Integer  exit_status = 0;
  double   x, y;
  NagError fail;

  INIT_FAIL(fail);

  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif
  printf("nag_bessel_i1 (s18afc) 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_i1 (s18afc).
       * Modified Bessel function I_1(x)
       */
      y = nag_bessel_i1(x, &fail);
      if (fail.code != NE_NOERROR)
        {
          printf("Error from nag_bessel_i1 (s18afc).\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_i1 (s18afc) Example Program Data
             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_i1 (s18afc) Example Program Results
     x           y
   0.000e+00   0.000e+00
   5.000e-01   2.579e-01
   1.000e+00   5.652e-01
   3.000e+00   3.953e+00
   6.000e+00   6.134e+01
   8.000e+00   3.999e+02
   1.000e+01   2.671e+03
   1.500e+01   3.281e+05
   2.000e+01   4.245e+07
  -1.000e+00  -5.652e-01
```


Example Program
Returned Values for the Bessel Function $I_1(x)$
10
8
6
4
2
0
$I_1(x)$
0
0.5
1
1.5
2
2.5
3
3.5
4
x