NAG C Library Function Document

nag_complex_hankel (s17dlc)

1 Purpose

nag_complex_hankel (s17dlc) returns a sequence of values for the Hankel functions $H^{(1)}_{\nu+n}(z)$ or $H^{(2)}_{\nu+n}(z)$ for complex z, non-negative ν and $n = 0, 1, \ldots, N-1$, with an option for exponential scaling.

2 Specification

```
void nag_complex_hankel (Integer m, double fnu, Complex z, Integer n,
     Nag_ScaleResType scal, Complex cy[], Integer *nz, NagError *fail)
```

3 Description

nag_complex_hankel (s17dlc) evaluates a sequence of values for the Hankel function $H^{(1)}_{\nu}(z)$ or $H^{(2)}_{\nu}(z)$, where z is complex, $-\pi < \arg z \leq \pi$, and ν is the real, non-negative order. The N-member sequence is generated for orders ν, $\nu+1, \ldots, \nu+N-1$. Optionally, the sequence is scaled by the factor e^{-iz} if the function is $H^{(1)}_{\nu}(z)$ or by the factor e^{iz} if the function is $H^{(2)}_{\nu}(z)$.

Note: although the function may not be called with ν less than zero, for negative orders the formulae $H^{(1)}_{-\nu}(z) = e^{\nu\pi i}H^{(1)}_{\nu}(z)$, and $H^{(2)}_{-\nu}(z) = e^{-\nu\pi i}H^{(2)}_{\nu}(z)$ may be used.

The function is derived from the routine CBESH in Amos (1986). It is based on the relation

$$H^{(m)}_{\nu}(z) = \frac{1}{p}e^{-p\nu}K_{\nu}(ze^{-p}),$$

where $p = \frac{i\pi}{2}$ if $m = 1$ and $p = -\frac{i\pi}{2}$ if $m = 2$, and the Bessel function $K_{\nu}(z)$ is computed in the right half-plane only. Continuation of $K_{\nu}(z)$ to the left half-plane is computed in terms of the Bessel function $I_{\nu}(z)$. These functions are evaluated using a variety of different techniques, depending on the region under consideration.

When N is greater than 1, extra values of $H^{(m)}_{\nu}(z)$ are computed using recurrence relations.

For very large $|z|$ or $(\nu+N-1)$, argument reduction will cause total loss of accuracy, and so no computation is performed. For slightly smaller $|z|$ or $(\nu+N-1)$, the computation is performed but results are accurate to less than half of ***machine precision***. If $|z|$ is very small, near the machine underflow threshold, or $(\nu+N-1)$ is too large, there is a risk of overflow and so no computation is performed. In all the above cases, a warning is given by the function.

4 References

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

Amos D E (1986) Algorithm 644: A portable package for Bessel functions of a complex argument and non-negative order *ACM Trans. Math. Software* **12** 265–273

5 Parameters

1: **m** – Integer *Input*

On entry: the kind of functions required.

If **m** $= 1$, the functions are $H^{(1)}_{\nu}(z)$.

If $\mathbf{m} = 2$, the functions are $H_\nu^{(2)}(z)$.

Constraint: $\mathbf{m} = 1$ or $\mathbf{m} = 2$.

2: **fnu** – double *Input*

On entry: the order, ν, of the first member of the sequence of functions.

Constraint: $\mathbf{fnu} \geq 0.0$.

3: **z** – Complex *Input*

On entry: the argument z of the functions.

Constraint: $\mathbf{z} \neq (0.0,\ 0.0)$.

4: **n** – Integer *Input*

On entry: the number, N, of members required in the sequence $H_\nu^{(\mathbf{m})}(z), H_{\nu+1}^{(\mathbf{m})}(z), \ldots, H_{\nu+N-1}^{(\mathbf{m})}(z)$.

Constraint: $\mathbf{n} \geq 1$.

5: **scal** – Nag_ScaleResType *Input*

On entry: the scaling option.

If **scal** = **Nag_UnscaleRes**, the results are returned unscaled.

If **scal** = **Nag_ScaleRes**, the results are returned scaled by the factor e^{-iz} when $\mathbf{m} = 1$, or by the factor e^{iz} when $\mathbf{m} = 2$.

Constraint: **scal** = **Nag_UnscaleRes** or **Nag_ScaleRes**.

6: **cy**[**n**] – Complex *Output*

On exit: the N required function values: $\mathbf{cy}[i-1]$ contains $H_{\nu+i-1}^{(\mathbf{m})}(z)$, for $i = 1, 2, \ldots, N$.

7: **nz** – Integer * *Output*

On exit: the number of components of **cy** that are set to zero due to underflow. If $\mathbf{nz} > 0$, then if $\mathrm{Im}\, z > 0.0$ and $\mathbf{m} = 1$, or $\mathrm{Im}\, z < 0.0$ and $\mathbf{m} = 2$, elements $\mathbf{cy}[0], \mathbf{cy}[1], \ldots, \mathbf{cy}[\mathbf{nz} - 1]$ are set to zero. In the complementary half-planes, **nz** simply states the number of underflows, and not which elements they are.

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

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

On entry, **m** has illegal value: $\mathbf{m} = \langle value \rangle$.

On entry, $\mathbf{n} = \langle value \rangle$.
Constraint: $\mathbf{n} \geq 1$.

NE_COMPLEX_ZERO

On entry, $\mathbf{z} = (0.0, 0.0)$.

NE_OVERFLOW_LIKELY

No computation because $\mathrm{abs}(\mathbf{z}) = \langle value \rangle < \langle value \rangle$.

No computation because $\mathbf{fnu} + \mathbf{n} - 1 = \langle value \rangle$ is too large.

NE_REAL

On entry, **fnu** $= \langle value \rangle$.
Constraint: **fnu** ≥ 0.

NE_TERMINATION_FAILURE

No computation – algorithm termination condition not met.

NE_TOTAL_PRECISION_LOSS

No computation because abs(**z**) $= \langle value \rangle > \langle value \rangle$.

No computation because **fnu** $+$ **n** $- 1 = \langle value \rangle > \langle value \rangle$.

NW_SOME_PRECISION_LOSS

Results lack precision, **fnu** $+$ **n** $- 1 = \langle value \rangle > \langle value \rangle$.

Results lack precision because abs(**z**) $= \langle value \rangle > \langle value \rangle$.

NE_BAD_PARAM

On entry, parameter $\langle value \rangle$ had an illegal value.

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 consult NAG for assistance.

7 Accuracy

All constants in nag_complex_hankel (s17dlc) are given to approximately 18 digits of precision. Calling the number of digits of precision in the floating-point arithmetic being used t, 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 nag_complex_hankel (s17dlc), the actual number of correct digits is limited, in general, by $p - s$, where $s \approx \max(1, |\log_{10}|z||, |\log_{10}\nu|)$ represents the number of digits lost due to the argument reduction. Thus the larger the values of $|z|$ and ν, the less the precision in the result. If nag_complex_hankel (s17dlc) is called with **n** > 1, then computation of function values via recurrence may lead to some further small loss of accuracy.

If function values which should nominally be identical are computed by calls to nag_complex_hankel (s17dlc) with different base values of ν and different **n**, the computed values may not agree exactly. Empirical tests with modest values of ν and z have shown that the discrepancy is limited to the least significant 3 – 4 digits of precision.

8 Further Comments

The time taken by the function for a call of nag_complex_hankel (s17dlc) is approximately proportional to the value of **n**, plus a constant. In general it is much cheaper to call nag_complex_hankel (s17dlc) with **n** greater than 1, rather than to make N separate calls to nag_complex_hankel (s17dlc).

Paradoxically, for some values of z and ν, it is cheaper to call nag_complex_hankel (s17dlc) with a larger value of **n** than is required, and then discard the extra function values returned. However, it is not possible to state the precise circumstances in which this is likely to occur. It is due to the fact that the base value used to start recurrence may be calculated in different regions for different **n**, and the costs in each region may differ greatly.

9 Example

The example program prints a caption and then proceeds to read sets of data from the input data stream. The first datum is a value for the kind of function, **m**, the second is a value for the order **fnu**, the third is a complex value for the argument, **z**, and the fourth is a character value used as a flag to set the parameter

scal. The program calls the function with $\mathbf{n} = 2$ to evaluate the function for orders **fnu** and **fnu** $+ 1$, and it prints the results. The process is repeated until the end of the input data stream is encountered.

9.1 Program Text

```
/* nag_complex_hankel (s17dlc) Example Program
 *
 * Copyright 2002 Numerical Algorithms Group.
 *
 * Mark 7, 2002.
 */

#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nags.h>

int main(void)
{
  Complex z, cy[2];
  double fnu;
  const Integer n = 2;
  Integer m, nz;
  Nag_ScaleResType scal_enum;
  char scal;

  Integer exit_status = EXIT_SUCCESS;
  NagError fail;

  INIT_FAIL(fail);

  /* Skip heading in data file */
  Vscanf("%*[^\n]");
  Vprintf("s17dlc Example Program Results\n");
  Vprintf("Calling with n = %ld\n", n);
  Vprintf("  m    fnu           z         scal         cy[0]              cy[1]
nz\n");
  while (scanf(" %ld %lf (%lf,%lf) ’%c’%*[^\n] ", &m, &fnu, &z.re, &z.im, &scal)
!= EOF)
    {
      /* Convert scal character to enum */
      if (scal == ’s’)
        {
          scal_enum = Nag_ScaleRes;
        }
      else if (scal == ’u’)
        {
          scal_enum = Nag_UnscaleRes;
        }
      else
        {
          Vprintf("Unrecognised character for Nag_ScaleResType type\n");
          exit_status = -1;
          goto END;
        }
      s17dlc(m, fnu, z, n, scal_enum, cy, &nz, &fail);
      if (fail.code == NE_NOERROR)
         Vprintf("  %ld %7.4f  (%7.3f,%7.3f)  ’%c’  (%7.3f,%7.3f)  (%7.3f,%7.3f)
%ld\n",
                m, fnu, z.re, z.im, scal, cy[0].re, cy[0].im, cy[1].re, cy[1].im,
nz);
      else
        {
          Vprintf("Error from s17dlc.\n%s\n", fail.message);
          exit_status = 1;
          goto END;
        }
    }
 END:
  return exit_status;
```

```
}
```

9.2 Program Data

```
s17dlc Example Program Data
 1    0.00    ( 0.3,  0.4)    'u'
 1    2.30    ( 2.0,  0.0)    'u'
 1    2.12    (-1.0,  0.0)    'u'
 2    6.00    ( 3.1, -1.6)    'u'
 2    6.00    ( 3.1, -1.6)    's'  - Values of m, fnu, z and scal
```

9.3 Program Results

```
s17dlc Example Program Results
Calling with n = 2
  m    fnu            z          scal         cy[0]              cy[1]      nz
  1  0.0000  (  0.300,  0.400)  'u'  (  0.347, -0.559)  ( -0.791, -0.818)  0
  1  2.3000  (  2.000,  0.000)  'u'  (  0.272, -0.740)  (  0.089, -1.412)  0
  1  2.1200  ( -1.000,  0.000)  'u'  ( -0.772, -1.693)  (  2.601,  6.527)  0
  2  6.0000  (  3.100, -1.600)  'u'  ( -1.371, -1.280)  ( -1.491, -5.993)  0
  2  6.0000  (  3.100, -1.600)  's'  (  7.050,  6.052)  (  8.614, 29.352)  0
```