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 \( \nu \) and \( n = 0, 1, \ldots, N - 1 \), with an option for exponential scaling.

2 Specification

```c
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 \( \nu \) is the real, non-negative order. The \( N \)-member sequence is generated for orders \( \nu, \nu+1, \ldots, \nu+N-1 \). 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 \( \nu \) 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

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

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


5 Parameters

1: \( m \) – Integer

    *Input*

    On entry: the kind of functions required.

    If \( m = 1 \), the functions are \( H^{(1)}_{\nu}(z) \).
If $m = 2$, the functions are $H^{(2)}_\nu(z)$.

Constraint: $m = 1$ or $m = 2$.

2: fnu – double

On entry: the order, $\nu$, of the first member of the sequence of functions.

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

3: z – Complex

On entry: the argument $z$ of the functions.

Constraint: $z \neq (0.0, 0.0)$.

4: n – Integer

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

Constraint: $n \geq 1$.

5: scal – Nag_ScaleResType

On entry: the scaling option.

If $\text{scal} = \text{Nag_UnscaleRes}$, the results are returned unscaled.

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

Constraint: $\text{scal} = \text{Nag_UnscaleRes}$ or $\text{Nag_ScaleRes}$.

6: cy[n] – Complex

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

7: nz – Integer *

On exit: the number of components of cy that are set to zero due to underflow. If $\text{nz} > 0$, then if $\text{Im} \cdot z > 0.0$ and $m = 1$, or $\text{Im} \cdot z < 0.0$ and $m = 2$, elements $\text{cy}[0], \text{cy}[1], \ldots, \text{cy}[\text{nz} - 1]$ 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 *

The NAG error parameter (see the Essential Introduction).

6  Error Indicators and Warnings

NE_INT

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

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

Constraint: $n \geq 1$.

NE_COMPLEX_ZERO

On entry, $z = (0.0, 0.0)$.

NE_OVERFLOW_LIKELY

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

No computation because $\text{fnu} + n - 1 = \langle\text{value}\rangle$ is too large.
On entry, \( fnu = (\text{value}). \)
Constraint: \( fnu \geq 0. \)

No computation – algorithm termination condition not met.

No computation because \( |z| = (\text{value}) > (\text{value}). \)
No computation because \( fnu + n - 1 = (\text{value}) > (\text{value}). \)

Results lack precision, \( fnu + n - 1 = (\text{value}) > (\text{value}). \)
Results lack precision because \( |z| = (\text{value}) > (\text{value}). \)

On entry, parameter \( (\text{value}) \) had an illegal value.

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.

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 \( s \approx \max(1, \lfloor \log_{10}(|z|) \rfloor, \lfloor \log_{10}(|\nu|) \rfloor) \) represents the number of digits lost due to the argument reduction. Thus the larger the values of \( |z| \) and \( \nu \), 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 \( \nu \) and different \( n \), the computed values may not agree exactly. Empirical tests with modest values of \( \nu \) and \( z \) have shown that the discrepancy is limited to the least significant 3 – 4 digits of precision.

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 \( \nu \), 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.

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

9.1 Program Text

```c
#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);

    Vscanf("%*
[^
]\n");
    Vprintf("s17dlc Example Program Results
");
    Vprintf("Calling with n = %ld
", n);
    Vprintf(" m fnu z scal cy[0] cy[1] nz
");

    while (scanf(" %ld %lf (%lf,%lf) '%c'% *[^
]\n");
    {
        /* 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
", 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", fail.message);
        exit_status = 1;
        goto END;

END:
return exit_status;
```

s17dlc

NAG C Library Manual
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

<table>
<thead>
<tr>
<th>m</th>
<th>fnu</th>
<th>z</th>
<th>scal</th>
<th>cy[0]</th>
<th>cy[1]</th>
<th>nz</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0.0000</td>
<td>( 0.300, 0.400) 'u'</td>
<td>( 0.347, -0.559)</td>
<td>( -0.791, -0.818)</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 2.3000</td>
<td>( 2.000, 0.000) 'u'</td>
<td>( 0.272, -0.740)</td>
<td>( 0.089, -1.412)</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 2.1200</td>
<td>( -1.000, 0.000) 'u'</td>
<td>( -0.772, -1.693)</td>
<td>( 2.601, 6.527)</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 6.0000</td>
<td>( 3.100, -1.600) 'u'</td>
<td>( -1.371, -1.280)</td>
<td>( -1.491, -5.993)</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 6.0000</td>
<td>( 3.100, -1.600) 's'</td>
<td>( 7.050, 6.052)</td>
<td>( 8.614, 29.352)</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>