NAG Toolbox

nag_specfun_bessel_j_complex (s17de)

1 Purpose

nag_specfun_bessel_j_complex (s17de) returns a sequence of values for the Bessel functions $J_{\nu+n}(z)$ for complex z, non-negative ν and $n = 0, 1, \ldots, N-1$, with an option for exponential scaling.

2 Syntax

```
[cy, nz, ifail] = nag_specfun_bessel_j_complex(fnu, z, n, scal)
[cy, nz, ifail] = s17de(fnu, z, n, scal)
```

3 Description

nag_specfun_bessel_j_complex (s17de) evaluates a sequence of values for the Bessel function $J_\nu(z)$, 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 $e^{-|\mathrm{Im}(z)|}$.

Note: although the function may not be called with ν less than zero, for negative orders the formula $J_{-\nu}(z) = J_\nu(z)\cos(\pi\nu) - Y_\nu(z)\sin(\pi\nu)$ may be used (for the Bessel function $Y_\nu(z)$, see nag_specfun_bessel_y_complex (s17dc)).

The function is derived from the function CBESJ in Amos (1986). It is based on the relations $J_\nu(z) = e^{\nu\pi i/2} I_\nu(-iz)$, $\mathrm{Im}(z) \geq 0.0$, and $J_\nu(z) = e^{-\nu\pi i/2} I_\nu(iz)$, $\mathrm{Im}(z) < 0.0$.

The Bessel function $I_\nu(z)$ is computed using a variety of techniques depending on the region under consideration.

When N is greater than 1, extra values of $J_\nu(z)$ 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 $\mathrm{Im}(z)$ is 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

5.1 Compulsory Input Parameters

1: **fnu** – REAL (KIND=nag_wp)

ν, the order of the first member of the sequence of functions.

Constraint: **fnu** ≥ 0.0.

2: **z** – COMPLEX (KIND=nag_wp)

The argument z of the functions.

3: **n** – INTEGER

N, the number of members required in the sequence $J_\nu(z), J_{\nu+1}(z), \ldots, J_{\nu+N-1}(z)$.

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

4: **scal** – CHARACTER(1)

The scaling option.

scal = 'U'
The results are returned unscaled.

scal = 'S'
The results are returned scaled by the factor $e^{-|\mathrm{Im}(z)|}$.

Constraint: **scal** = 'U' or 'S'.

5.2 Optional Input Parameters

None.

5.3 Output Parameters

1: **cy**(**n**) – COMPLEX (KIND=nag_wp) array

The N required function values: **cy**(i) contains $J_{\nu+i-1}(z)$, for $i = 1, 2, \ldots, N$.

2: **nz** – INTEGER

The number of components of **cy** that are set to zero due to underflow. If $\mathbf{nz} > 0$, then elements $\mathbf{cy}(\mathbf{n} - \mathbf{nz} + 1)$, $\mathbf{cy}(\mathbf{n} - \mathbf{nz} + 2), \ldots, \mathbf{cy}(\mathbf{n})$ are set to zero.

3: **ifail** – INTEGER

ifail $= 0$ unless the function detects an error (see Section 5).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= 1$

On entry, **fnu** < 0.0,
or **n** < 1,
or **scal** $\neq$ 'U' or 'S'.

ifail $= 2$

No computation has been performed due to the likelihood of overflow, because Im **z** is larger than a machine-dependent threshold value. This error exit can only occur when **scal** = 'U'.

ifail $= 3$ (*warning*)

The computation has been performed, but the errors due to argument reduction in elementary functions make it likely that the results returned by nag_specfun_bessel_j_complex (s17de) are accurate to less than half of ***machine precision***. This error exit may occur if either abs(**z**) or $\mathbf{fnu} + \mathbf{n} - 1$ is greater than a machine-dependent threshold value.

ifail $= 4$

No computation has been performed because the errors due to argument reduction in elementary functions mean that all precision in results returned by nag_specfun_bessel_j_complex (s17de) would be lost. This error exit may occur when either abs(**z**) or $\mathbf{fnu} + \mathbf{n} - 1$ is greater than a machine-dependent threshold value.

ifail $= 5$

No results are returned because the algorithm termination condition has not been met. This may occur because the arguments supplied to nag_specfun_bessel_j_complex (s17de) would have caused overflow or underflow.

ifail $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

ifail $= -399$

Your licence key may have expired or may not have been installed correctly.

ifail $= -999$

Dynamic memory allocation failed.

7 Accuracy

All constants in nag_specfun_bessel_j_complex (s17de) 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_specfun_bessel_j_complex (s17de), 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_specfun_bessel_j_complex (s17de) is called with $\mathbf{n} > 1$, 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_specfun_bessel_j_complex (s17de) 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 for a call of nag_specfun_bessel_j_complex (s17de) is approximately proportional to the value of **n**, plus a constant. In general it is much cheaper to call nag_specfun_bessel_j_complex (s17de) with **n** greater than 1, rather than to make N separate calls to nag_specfun_bessel_j_complex (s17de).

Paradoxically, for some values of z and ν, it is cheaper to call nag_specfun_bessel_j_complex (s17de) 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.

Note that if the function required is $J_0(x)$ or $J_1(x)$, i.e., $\nu = 0.0$ or 1.0, where x is real and positive, and only a single unscaled function value is required, then it may be much cheaper to call nag_specfun_bessel_j0_real (s17ae) or nag_specfun_bessel_j1_real (s17af) respectively.

9 Example

This example prints a caption and then proceeds to read sets of data from the input data stream. The first datum is a value for the order **fnu**, the second is a complex value for the argument, **z**, and the third is a character value to set the argument **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

```
function s17de_example

fprintf('s17de example results\n\n');

n  = nag_int(2);
nu = [0             2.3       2.12      1.58          1.58];
z  = [0.3 + 0.4i;   2 + 0i;  -1 + 0i;  -2.3 + 5.6i;  -2.3 + 5.6i];
scal = {'U';        'U';      'U';      'U';          'S'};

fprintf('   nu           z         scaled?');
fprintf('     J_{nu+%d}(z) ',[0:n-1]);
fprintf('   nz\n');
for i=1:numel(nu)

  [cy, nz, ifail] = s17de(nu(i), complex(z(i)), n, scal{i});

  fprintf('%7.3f  %7.3f%+7.3fi', nu(i), real(z(i)), imag(z(i)));
  if scal{i} == 'U'
     fprintf('  unscaled');
  else
     fprintf('    scaled');
  end
  for j = 1:n
    fprintf(' %7.3f%+8.3fi', real(cy(j)), imag(cy(j)));
  end
  fprintf('%3d\n',nz);
end
```

9.2 Program Results

```
s17de example results

   nu           z         scaled?     J_{nu+0}(z)      J_{nu+1}(z)    nz
  0.000    0.300 +0.400i  unscaled   1.017  -0.061i   0.157  +0.197i  0
  2.300    2.000 +0.000i  unscaled   0.272  -0.000i   0.089  -0.000i  0
  2.120   -1.000 +0.000i  unscaled   0.088  +0.035i  -0.014  -0.006i  0
  1.580   -2.300 +5.600i  unscaled  -1.551 -36.476i  25.910  +2.677i  0
  1.580   -2.300 +5.600i    scaled  -0.006  -0.135i   0.096  +0.010i  0
```