NAG Library Routine Document

S18DCF

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details.

1 Purpose

S18DCF returns a sequence of values for the modified Bessel functions $K_{\nu+n}(z)$ for complex z, non-negative ν and $n = 0, 1, \ldots, N-1$, with an option for exponential scaling.

2 Specification

```
SUBROUTINE S18DCF (FNU, Z, N, SCAL, CY, NZ, IFAIL)

INTEGER                 N, NZ, IFAIL
REAL (KIND=nag_wp)      FNU
COMPLEX (KIND=nag_wp) Z, CY(N)
CHARACTER(1)            SCAL
```

3 Description

S18DCF evaluates a sequence of values for the modified Bessel function $K_\nu(z)$, 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^z.

The routine is derived from the routine CBESK in Amos (1986).

Note: although the routine may not be called with ν less than zero, for negative orders the formula $K_{-\nu}(z) = K_\nu(z)$ may be used.

When N is greater than 1, extra values of $K_\nu(z)$ 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 routine.

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: FNU – REAL (KIND=nag_wp) *Input*

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

Constraint: FNU ≥ 0.0.

2: Z – COMPLEX (KIND=nag_wp) *Input*

On entry: the argument z of the functions.

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

3: N – INTEGER *Input*

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

Constraint: $N \geq 1$.

4: SCAL – CHARACTER(1) *Input*

On entry: the scaling option.

SCAL = 'U'
The results are returned unscaled.

SCAL = 'S'
The results are returned scaled by the factor e^{z}.

Constraint: SCAL = 'U' or 'S'.

5: CY(N) – COMPLEX (KIND=nag_wp) array *Output*

On exit: the N required function values: CY(i) contains $K_{\nu+i-1}(z)$, for $i = 1, 2, \ldots, N$.

6: NZ – INTEGER *Output*

On exit: the number of components of CY that are set to zero due to underflow. If NZ > 0 and Re(z) ≥ 0.0, elements CY(1), CY(2), $\ldots$, CY(NZ) are set to zero. If Re(z) < 0.0, NZ simply states the number of underflows, and not which elements they are.

7: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this parameter you should refer to Section 3.3 in the Essential Introduction for details.

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, if you are not familiar with this parameter, the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

On exit: IFAIL = 0 unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

If on entry IFAIL = 0 or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

IFAIL = 1

On entry, FNU > 0.0,
or Z = $(0.0, 0.0)$,
or N < 1,
or SCAL $\neq$ 'U' or 'S'.

IFAIL = 2

No computation has been performed due to the likelihood of overflow, because abs(Z) is less than a machine-dependent threshold value (given in the Users' Note for your implementation).

IFAIL = 3

No computation has been performed due to the likelihood of overflow, because FNU + N − 1 is too large – how large depends on Z and the overflow threshold of the machine.

IFAIL $= 4$

The computation has been performed, but the errors due to argument reduction in elementary functions make it likely that the results returned by S18DCF are accurate to less than half of ***machine precision***. This error exit may occur if either abs(Z) or FNU $+$ N $-$ 1 is greater than a machine-dependent threshold value (given in the Users' Note for your implementation).

IFAIL $= 5$

No computation has been performed because the errors due to argument reduction in elementary functions mean that all precision in results returned by S18DCF would be lost. This error exit may occur when either abs(Z) or FNU $+$ N $-$ 1 is greater than a machine-dependent threshold value (given in the Users' Note for your implementation).

IFAIL $= 6$

No results are returned because the algorithm termination condition has not been met. This may occur because the parameters supplied to S18DCF would have caused overflow or underflow.

7 Accuracy

All constants in S18DCF 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 S18DCF, 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 S18DCF 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 S18DCF 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 S18DCF is approximately proportional to the value of N, plus a constant. In general it is much cheaper to call S18DCF with N greater than 1, rather than to make N separate calls to S18DCF.

Paradoxically, for some values of z and ν, it is cheaper to call S18DCF 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 $K_0(x)$ or $K_1(x)$, i.e., $\nu = 0.0$ or 1.0, where x is real and positive, and only a single function value is required, then it may be much cheaper to call S18ACF, S18ADF, S18CCF or S18CDF, depending on whether a scaled result is required or not.

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 order FNU, the second is a complex value for the argument, Z, and the third is a character value to set the parameter SCAL. The program calls the routine with 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

```
    Program s18dcfe

!     S18DCF Example Program Text

!     Mark 24 Release. NAG Copyright 2012.

!     .. Use Statements ..
      Use nag_library, Only: nag_wp, s18dcf
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: n = 2, nin = 5, nout = 6
!     .. Local Scalars ..
      Complex (Kind=nag_wp)            :: z
      Real (Kind=nag_wp)               :: fnu
      Integer                          :: ifail, ioerr, nz
      Character (1)                    :: scal
!     .. Local Arrays ..
      Complex (Kind=nag_wp)            :: cy(n)
!     .. Executable Statements ..
      Write (nout,*) 'S18DCF Example Program Results'

!     Skip heading in data file
      Read (nin,*)

      Write (nout,*)
      Write (nout,99999) 'Calling with N =', n
      Write (nout,*)
      Write (nout,*) &
        '   FNU             Z        SCAL       CY(1)               CY(2)', &
        '        NZ'
      Write (nout,*)

data: Do
        Read (nin,*,Iostat=ioerr) fnu, z, scal

        If (ioerr<0) Then
          Exit data
        End If

        ifail = 0
        Call s18dcf(fnu,z,n,scal,cy,nz,ifail)

        Write (nout,99998) fnu, z, scal, cy(1), cy(2), nz
      End Do data

99999 Format (1X,A,I2)
99998 Format (1X,F7.4,'  (',F7.3,',',F7.3,')   ',A,2('  (',F7.3,',',F7.3,')'), &
        I4)
    End Program s18dcfe
```

9.2 Program Data

```
S18DCF Example Program Data
 0.00    ( 0.3,  0.4)    'U'
 2.30    ( 2.0,  0.0)    'U'
 2.12    (-1.0,  0.0)    'U'
 5.10    ( 3.0,  2.0)    'U'
 5.10    ( 3.0,  2.0)    'S'
```

9.3 Program Results

```
 S18DCF Example Program Results

 Calling with N = 2

    FNU             Z        SCAL       CY(1)               CY(2)         NZ
```

```
  0.0000  (   0.300,  0.400)   U  (   0.831, -0.803)  (   0.831, -1.735)   0
  2.3000  (   2.000,  0.000)   U  (   0.325,  0.000)  (   0.909,  0.000)   0
  2.1200  ( -1.000,  0.000)   U  (   1.763, -1.047)  ( -8.087,  3.147)   0
  5.1000  (   3.000,  2.000)   U  ( -0.426,  0.243)  ( -0.810,  1.255)   0
  5.1000  (   3.000,  2.000)   S  ( -0.880, -9.803)  (-16.150,-25.293)   0
```