

## NAG Library Routine Document

### S18AQF

**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

S18AQF returns an array of values of the modified Bessel function  $K_0(x)$ .

## 2 Specification

```
SUBROUTINE S18AQF (N, X, F, IVALID, IFAIL)
  INTEGER          N, IVALID(N), IFAIL
  REAL (KIND=nag_wp) X(N), F(N)
```

## 3 Description

S18AQF evaluates an approximation to the modified Bessel function of the second kind  $K_0(x_i)$  for an array of arguments  $x_i$ , for  $i = 1, 2, \dots, n$ .

**Note:**  $K_0(x)$  is undefined for  $x \leq 0$  and the routine will fail for such arguments.

The routine is based on five Chebyshev expansions:

For  $0 < x \leq 1$ ,

$$K_0(x) = -\ln x \sum_{r=0} a_r T_r(t) + \sum_{r=0} b_r T_r(t), \quad \text{where } t = 2x^2 - 1.$$

For  $1 < x \leq 2$ ,

$$K_0(x) = e^{-x} \sum_{r=0} c_r T_r(t), \quad \text{where } t = 2x - 3.$$

For  $2 < x \leq 4$ ,

$$K_0(x) = e^{-x} \sum_{r=0} d_r T_r(t), \quad \text{where } t = x - 3.$$

For  $x > 4$ ,

$$K_0(x) = \frac{e^{-x}}{\sqrt{x}} \sum_{r=0} e_r T_r(t), \quad \text{where } t = \frac{9-x}{1+x}.$$

For  $x$  near zero,  $K_0(x) \simeq -\gamma - \ln\left(\frac{x}{2}\right)$ , where  $\gamma$  denotes Euler's constant. This approximation is used when  $x$  is sufficiently small for the result to be correct to *machine precision*.

For large  $x$ , where there is a danger of underflow due to the smallness of  $K_0$ , the result is set exactly to zero.

## 4 References

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

## 5 Arguments

- 1: N – INTEGER *Input*  
*On entry:*  $n$ , the number of points.  
*Constraint:*  $N \geq 0$ .
- 2: X(N) – REAL (KIND=nag\_wp) array *Input*  
*On entry:* the argument  $x_i$  of the function, for  $i = 1, 2, \dots, N$ .  
*Constraint:*  $X(i) > 0.0$ , for  $i = 1, 2, \dots, N$ .
- 3: F(N) – REAL (KIND=nag\_wp) array *Output*  
*On exit:*  $K_0(x_i)$ , the function values.
- 4: IVALID(N) – INTEGER array *Output*  
*On exit:* IVALID( $i$ ) contains the error code for  $x_i$ , for  $i = 1, 2, \dots, N$ .  
 IVALID( $i$ ) = 0  
     No error.  
 IVALID( $i$ ) = 1  
      $x_i \leq 0.0$ ,  $K_0(x_i)$  is undefined. F( $i$ ) contains 0.0.
- 5: IFAIL – INTEGER *Input/Output*  
*On entry:* IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this argument you should refer to Section 3.4 in How to Use the NAG Library and its Documentation 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 argument, 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, at least one value of X was invalid.  
 Check IVALID for more information.

IFAIL = 2

On entry,  $N = \langle value \rangle$ .  
 Constraint:  $N \geq 0$ .

IFAIL = -99

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

See Section 3.9 in How to Use the NAG Library and its Documentation for further information.

IFAIL = -399

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

See Section 3.8 in How to Use the NAG Library and its Documentation for further information.

IFAIL = -999

Dynamic memory allocation failed.

See Section 3.7 in How to Use the NAG Library and its Documentation for further information.

## 7 Accuracy

Let  $\delta$  and  $\epsilon$  be the relative errors in the argument and result respectively.

If  $\delta$  is somewhat larger than the *machine precision* (i.e., if  $\delta$  is due to data errors etc.), then  $\epsilon$  and  $\delta$  are approximately related by:

$$\epsilon \simeq \left| \frac{xK_1(x)}{K_0(x)} \right| \delta.$$

Figure 1 shows the behaviour of the error amplification factor

$$\left| \frac{xK_1(x)}{K_0(x)} \right|.$$

However, if  $\delta$  is of the same order as *machine precision*, then rounding errors could make  $\epsilon$  slightly larger than the above relation predicts.

For small  $x$ , the amplification factor is approximately  $\left| \frac{1}{\ln x} \right|$ , which implies strong attenuation of the error, but in general  $\epsilon$  can never be less than the *machine precision*.

For large  $x$ ,  $\epsilon \simeq x\delta$  and we have strong amplification of the relative error. Eventually  $K_0$ , which is asymptotically given by  $\frac{e^{-x}}{\sqrt{x}}$ , becomes so small that it cannot be calculated without underflow and hence the routine will return zero. Note that for large  $x$  the errors will be dominated by those of the standard function exp.

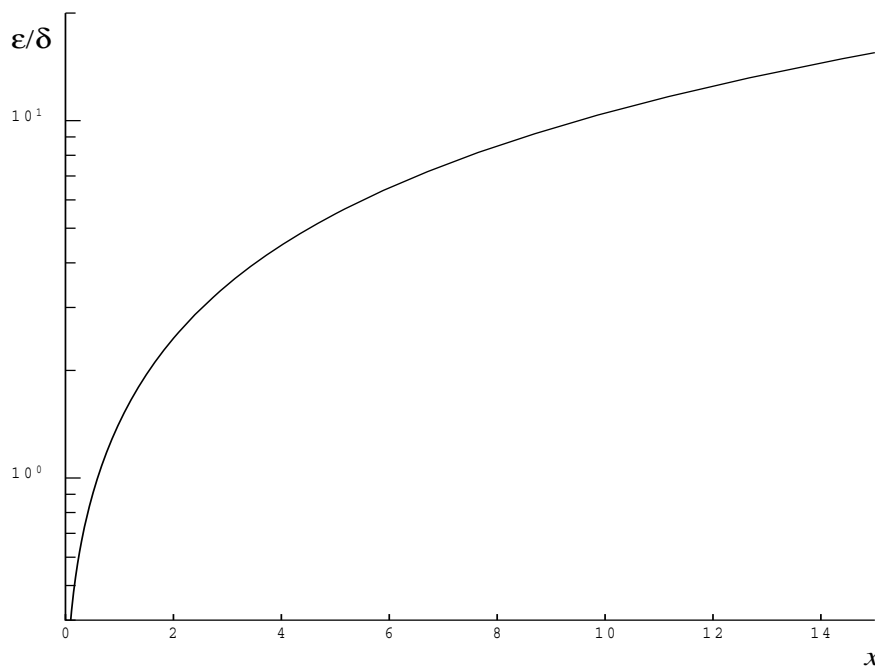


Figure 1

## 8 Parallelism and Performance

S18AQF is not threaded in any implementation.

## 9 Further Comments

None.

## 10 Example

This example reads values of X from a file, evaluates the function at each value of  $x_i$  and prints the results.

### 10.1 Program Text

```

Program s18aqfe

!      S18AQF Example Program Text

!      Mark 26 Release. NAG Copyright 2016.

!      .. Use Statements ..
      Use nag_library, Only: nag_wp, s18aqf
!      .. Implicit None Statement ..
      Implicit None
!      .. Parameters ..
      Integer, Parameter          :: nin = 5, nout = 6
!      .. Local Scalars ..
      Integer                     :: i, ifail, n
!      .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable :: f(:), x(:)
      Integer, Allocatable        :: ivalid(:)
!      .. Executable Statements ..
      Write (nout,*) 'S18AQF Example Program Results'

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

      Write (nout,*)
      Write (nout,*) '      X      F      IVALID'
      Write (nout,*)

      Read (nin,*) n

      Allocate (x(n),f(n),ivalid(n))

      Read (nin,*) x(1:n)

      ifail = 0
      Call s18aqf(n,x,f,ivalid,ifail)

      Do i = 1, n
         Write (nout,99999) x(i), f(i), ivalid(i)
      End Do

99999 Format (1X,1P,2E12.3,I5)
End Program s18aqfe

```

### 10.2 Program Data

S18AQF Example Program Data

10

0.4 0.6 1.4 1.6 2.5 3.5 6.0 8.0 10.0 1000.0

### **10.3 Program Results**

S18AQF Example Program Results

X	F	INVALID
4.000E-01	1.115E+00	0
6.000E-01	7.775E-01	0
1.400E+00	2.437E-01	0
1.600E+00	1.880E-01	0
2.500E+00	6.235E-02	0
3.500E+00	1.960E-02	0
6.000E+00	1.244E-03	0
8.000E+00	1.465E-04	0
1.000E+01	1.778E-05	0
1.000E+03	0.000E+00	0

---