Chapter X02

Machine Constants

Contents

1 Scope of the Chapter

This chapter is concerned with **parameters** which characterise certain aspects of the **computing environment** in which the NAG Fortran Library is implemented. They relate primarily to floating-point arithmetic, but also to integer arithmetic, the elementary functions and exception handling. The values of the parameters vary from one implementation of the Library to another, but within the context of a single implementation they are constants.

The parameters are intended for use primarily by other routines in the Library, but users of the Library may sometimes need to refer to them directly.

Each parameter-value is returned by a separate Fortran function. Because of the trivial nature of the functions, individual routine documents are not provided; the necessary details are given in Section 3 of this Introduction.

2 Background to the Problems

2.1 Floating-Point Arithmetic

2.1.1 A model of floating-point arithmetic

In order to characterise the important properties of floating-point arithmetic by means of a small number of parameters, NAG uses a simplified **model** of floating-point arithmetic. The parameters of the model can be chosen to provide a sufficiently close description of the behaviour of actual implementations of floating-point arithmetic, but not, in general, an exact description; actual implementations vary too much in the details of how numbers are represented or arithmetic operations are performed.

The model is based on that developed by Brown [1], but differs in some respects. The essential features are summarised here.

The model is characterised by four integer parameters and one logical parameter. The four integer parameters are:

b:	the base
p:	the precision (i.e., the number of significant base-b digits)
e_{min}:	the minimum exponent
e_{max}:	the maximum exponent

These parameters define a set of numerical values of the form:

$$f\times b^e$$

where the exponent e must lie in the range $[e_{\mathrm{min}},\ e_{\mathrm{max}}]$, and the fraction f (also called the mantissa or significand) lies in the range $[1/b, 1)$, and may be written:

$$f = 0.f_1f_2{\cdots}f_p$$

Thus f is a p-digit fraction to the base b; the f_i are the base-b digits of the fraction: they are integers in the range 0 to $b-1$, and the leading digit f_1 must not be zero.

The set of values so defined (together with zero) are called **model numbers.** For example, if $b = 10$, $p = 5$, $e_{\mathrm{min}} = -99$ and $e_{\mathrm{max}} = +99$, then a typical model number is 0.12345×10^{67}.

The model numbers must obey certain rules for the computed results of the following basic arithmetic operations: addition, subtraction, multiplication, negation, absolute value, and comparisons. The rules depend on the value of the logical parameter ROUNDS.

If ROUNDS is **true**, then the computed result must be the nearest model number to the exact result (assuming that overflow or underflow does not occur); if the exact result is midway between two model numbers, then it may be rounded either way.

If ROUNDS is **false**, then: if the exact result is a model number, the computed result must be equal to the exact result; otherwise, the computed result may be either of the adjacent model numbers on either side of the exact result.

For division and square root, this latter rule is further relaxed (regardless of the value of ROUNDS): the computed result may also be one of the next adjacent model numbers on either side of the permitted values just stated.

On some machines, the full set of representable floating-point numbers conforms to the rules of the model with appropriate values of b, p, e_{min}, e_{max} and ROUNDS. For example, for DEC VAX machines in single precision:

$$\begin{array}{lcr} b & = & 2 \\ p & = & 24 \\ e_{\mathrm{min}} & = & -127 \\ e_{\mathrm{max}} & = & 127 \end{array} \quad \text{and ROUNDS is } \textbf{true}.$$

For machines supporting IEEE binary double precision arithmetic:

$$\begin{array}{lcr} b & = & 2 \\ p & = & 53 \\ e_{\mathrm{min}} & = & -1021 \\ e_{\mathrm{max}} & = & 1024 \end{array} \quad \text{and ROUNDS is } \textbf{true}.$$

For other machines, values of the model parameters must be chosen which define a large subset of the representable numbers; typically it may be necessary to decrease p by 1 (in which case ROUNDS is always set to **false**), or to increase e_{min} or decrease e_{max} by a little bit. There are additional rules to ensure that arithmetic operations on those representable numbers that are not model numbers are consistent with arithmetic on model numbers.

(**Note.** The model used here differs from that described in Brown [1] in the following respects: square-root is treated, like division, as a weakly supported operator; and the logical parameter ROUNDS has been introduced to take account of machines with good rounding.)

2.1.2 Derived parameters of floating-point arithmetic

Most numerical algorithms require access, not to the basic parameters of the model, but to certain derived values, of which the most important are:

the ***machine precision*** ϵ:	$= \left(\frac{1}{2}\right) \times b^{1-p}$ if ROUNDS is **true**,
	$= b^{1-p}$ otherwise (but see Note below).
the smallest positive model number:	$= b^{e_{\mathrm{min}}-1}$
the largest positive model number:	$= (1 - b^{-p}) \times b^{e_{\mathrm{max}}}$

Note. This value is increased very slightly in some implementations to ensure that the computed result of $1 + \epsilon$ or $1 - \epsilon$ differs from 1. For example in IEEE binary single precision arithmetic the value is set to $2^{-24} + 2^{-47}$.

Two additional derived values are used in the NAG Fortran Library. Their definitions depend not only on the properties of the basic arithmetic operations just considered, but also on properties of some of the elementary functions. We define the **safe range** parameter to be the smallest positive model number z such that for any x in the range $[z, 1/z]$ the following can be computed without undue loss of accuracy, overflow, underflow or other error:

$-x$

$1/x$

$-1/x$

SQRT(x)

LOG(x)

EXP(LOG(x))

y**(LOG(x)/LOG(y)) for any y

In a similar fashion we define the safe range parameter for complex arithmetic as the smallest positive model number z such that for any x in the range $[z, 1/z]$ the following can be computed without any undue loss of accuracy, overflow, underflow or other error:

$-w$

$1/w$

$-1/w$

SQRT(w)

LOG(w)

EXP(LOG(w))

y**(LOG(w)/LOG(y)) for any y

ABS(w)

where w is any of x, ix, $x + ix$, $1/x$, i/x, $1/x + i/x$, and i is the square root of -1.

This parameter was introduced to take account of the quality of complex arithmetic on the machine. On machines with well implemented complex arithmetic, its value will differ from that of the real safe range parameter by a small multiplying factor less than 10. For poorly implemented complex arithmetic this factor may be larger by many orders of magnitude.

2.2 Other Aspects of the Computing Environment

No attempt has been made to characterise comprehensively any other aspects of the computing environment. The other functions in this chapter provide specific information that is occasionally required by routines in the Library.

3 Recommendations on Choice and Use of Available Routines

Note. Refer to the Users' Note for your implementation to check that a routine is available.

3.1 Historical Note

At Mark 12 a new set of routines was introduced to return parameters of floating-point arithmetic. The new set of routines is more carefully defined, and they do not require a dummy parameter. They are listed in Section 3.2. The older routines have since been withdrawn (see Section 4).

3.2 Parameters of Floating-point Arithmetic

real FUNCTION X02AJF()	returns the **machine precision**, i.e., $\left(\frac{1}{2}\right) \times b^{1-p}$ if ROUNDS is **true** or b^{1-p} otherwise (or a value very slightly larger than this, see Section 2.1.2)
real FUNCTION X02AKF()	returns the smallest positive model number, i.e., $b^{e_{\min}-1}$
real FUNCTION X02ALF()	returns the largest positive model number, i.e., $(1 - b^{-p}) \times b^{e_{\max}}$
real FUNCTION X02AMF()	returns the **safe range** parameter as defined in Section 2.1.2
real FUNCTION X02ANF()	returns the **safe range** parameter for complex arithmetic as defined in Section 2.1.2
INTEGER FUNCTION X02BHF()	returns the model parameter b
INTEGER FUNCTION X02BJF()	returns the model parameter p
INTEGER FUNCTION X02BKF()	returns the model parameter $e_{\min}$
INTEGER FUNCTION X02BLF()	returns the model parameter $e_{\max}$
LOGICAL FUNCTION X02DJF()	returns the model parameter ROUNDS

3.3 Parameters of Other Aspects of the Computing Environment

real FUNCTION X02AHF(X) ***real*** X	returns the largest positive **real** argument for which the intrinsic functions SIN and COS return a result with some meaningful accuracy
INTEGER FUNCTION X02BBF(X) ***real*** X	returns the largest positive integer value
INTEGER FUNCTION X02BEF(X) ***real*** X	returns the maximum number of decimal digits which can be accurately represented over the whole range of floating-point numbers
LOGICAL FUNCTION X02DAF(X) ***real*** X	returns **false** if the system sets underflowing quantities to zero, without any error indication or undesirable warning or system overhead

The parameter X in these routines is a dummy parameter.

4 Routines Withdrawn or Scheduled for Withdrawal

Since Mark 13 the following routines have been withdrawn. Advice on replacing calls to these routines is given in the document 'Advice on Replacement Calls for Withdrawn/Superseded Routines'.

X02AAF	X02ABF	X02ACF	X02ADF	X02AEF	X02AFF
X02AGF	X02BAF	X02BCF	X02BDF	X02CAF	

5 References

[1] Brown W S (1981) A simple but realistic model of floating-point computation *ACM Trans. Math. Software* **7** 445–480

6 Example Program

The example program listed below simply prints the values of all the functions in Chapter X02. Obviously the results will vary from one implementation of the Library to another. The results listed in Section 6.3 are those from a double precision implementation on a Silicon Graphics workstation.

6.1 Example Text

```
*     X02AJF Example Program Text
*     Mark 17 Revised.  NAG Copyright 1995.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. External Functions ..
      real             X02AHF, X02AJF, X02AKF, X02ALF, X02AMF, X02ANF
      INTEGER          X02BBF, X02BEF, X02BHF, X02BJF, X02BKF, X02BLF
      LOGICAL          X02DAF, X02DJF
      EXTERNAL         X02AHF, X02AJF, X02AKF, X02ALF, X02AMF, X02ANF,
     +                 X02BBF, X02BEF, X02BHF, X02BJF, X02BKF, X02BLF,
     +                 X02DAF, X02DJF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'X02AJF Example Program Results'
      WRITE (NOUT,*)
      WRITE (NOUT,*) '(results are machine-dependent)'
      WRITE (NOUT,*)
      WRITE (NOUT,*) 'The basic parameters of the model'
      WRITE (NOUT,*)
      WRITE (NOUT,99999) ' X02BHF = ', X02BHF(),
     +  '  (the model parameter B)'
```

```
      WRITE (NOUT,99999) ' X02BJF = ', X02BJF(),
     +  '  (the model parameter P)'
      WRITE (NOUT,99999) ' X02BKF = ', X02BKF(),
     +  '  (the model parameter EMIN)'
      WRITE (NOUT,99999) ' X02BLF = ', X02BLF(),
     +  '  (the model parameter EMAX)'
      WRITE (NOUT,99998) ' X02DJF = ', X02DJF(),
     +  '  (the model parameter ROUNDS)'
      WRITE (NOUT,*)
      WRITE (NOUT,*) 'Derived parameters of floating-point arithmetic'
      WRITE (NOUT,*)
      WRITE (NOUT,*) ' X02AJF = ', X02AJF(), '  (the machine precision)'
      WRITE (NOUT,*) ' X02AKF = ', X02AKF(),
     +  '  (the smallest positive model number)'
      WRITE (NOUT,*) ' X02ALF = ', X02ALF(),
     +  '  (the largest positive model number)'
      WRITE (NOUT,*) ' X02AMF = ', X02AMF(),
     +  '  (the real safe range parameter)'
      WRITE (NOUT,*) ' X02ANF = ', X02ANF(),
     +  '  (the complex safe range parameter)'
      WRITE (NOUT,*)
      WRITE (NOUT,*)
     +  'Parameters of other aspects of the computing environment'
      WRITE (NOUT,*)
      WRITE (NOUT,*) ' X02AHF = ', X02AHF(0.0e0),
     +  '  (largest argument for SIN and COS)'
      WRITE (NOUT,99997) ' X02BBF = ', X02BBF(0.0e0),
     +  '  (largest positive integer)'
      WRITE (NOUT,99997) ' X02BEF = ', X02BEF(0.0e0),
     +  '  (precision in decimal digits)'
      WRITE (NOUT,99996) ' X02DAF = ', X02DAF(0.0e0),
     +  '  (indicates how underflow is handled)'
      STOP
*
99999 FORMAT (1X,A,I7,A)
99998 FORMAT (1X,A,L7,A)
99997 FORMAT (1X,A,I20,A)
99996 FORMAT (1X,A,L20,A)
      END
```

6.2 Example Data

None.

6.3 Example Results

```
X02AJF Example Program Results

(results are machine-dependent)

The basic parameters of the model

 X02BHF =       2  (the model parameter B)
 X02BJF =      53  (the model parameter P)
 X02BKF =   -1021  (the model parameter EMIN)
 X02BLF =    1024  (the model parameter EMAX)
 X02DJF =       T  (the model parameter ROUNDS)
```

```
Derived parameters of floating-point arithmetic

 X02AJF =   1.1102230246251600E-16  (the machine precision)
 X02AKF =   2.2250738585072107-308  (the smallest positive model number)
 X02ALF =   1.7976931348623093+308  (the largest positive model number)
 X02AMF =   2.2250738585072107-308  (the real safe range parameter)
 X02ANF =   2.2250738585072107-308  (the complex safe range parameter)

Parameters of other aspects of the computing environment

 X02AHF =   1.8014398509481900E+16  (largest argument for SIN and COS)
 X02BBF =           2147483647  (largest positive integer)
 X02BEF =                   15  (precision in decimal digits)
 X02DAF =                    F  (indicates how underflow is handled)
```