NAG Library Chapter Introduction

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 C 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 functions in the Library, but users of the Library may sometimes need to refer to them directly.

Most of these constants are *not* functions, but they are defined in the header file <nagx02.h>. Defined constant names are specified in upper case characters, and functions in lower case. Those machine constants which are defined as functions have also been given upper case names using #define in <nagx02.h>.

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 (1981), but differs in some respects. The essential features are summarised here.

The model is characterised by four integer arguments. The four integer arguments 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 $[e_{\min}, e_{\max}]$, and the fraction f (also called the mantissa or significand) lies in the range $[1/b, 1)$, and may be written

$$f = 0.f_1 f_2 \cdots f_p$$

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 $b = 10$, $p = 5$, $e_{\min} = -99$ and $e_{\max} = +99$, 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 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.

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

On many machines, the full set of representable floating-point numbers conforms to the rules of the model with appropriate values of b, p, $e_{\min}$ and $e_{\max}$. For machines supporting IEEE binary double precision arithmetic:

$$\begin{aligned} b &= 2 \\ p &= 53 \\ e_{\text{min}} &= -1021 \\ e_{\text{max}} &= 1024. \end{aligned}$$

(**Note:** the model used here differs from that described in Brown (1981) in the following respect: square-root is treated, like division, as a weakly supported operator.)

2.1.2 Derived arguments 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: $= \left(\frac{1}{2}\right) \times b^{1-p}$
the smallest positive model number: $= b^{e_{\text{min}}-1}$
the largest positive model number: $= (1 - b^{-p}) \times b^{e_{\text{max}}}$

It is important to note that the machine precision defined here differs from that defined by ISO (1997).

Two additional derived values are used in the NAG C 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 argument 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

$|w|$

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.

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 functions in the Library.

3 Recommendations on Choice and Use of Available Functions

Derived parameters of model of floating-point arithmetic,
- largest positive model number ..... nag_real_largest_number (X02ALC)
- machine precision ..... nag_machine_precision (X02AJC)
- safe range ..... nag_real_safe_small_number (X02AMC)
- safe range of complex floating-point arithmetic ..... nag_complex_safe_small_number (X02ANC)
- smallest positive model number ..... nag_real_smallest_number (X02AKC)

Largest permissible argument for SIN and COS ..... nag_max_sine_argument (X02AHC)

Largest representable integer ..... nag_max_integer (X02BBC)

Maximum number of decimal digits that can be represented ..... nag_decimal_digits (X02BEC)

Parameters of model of floating-point arithmetic,
- b ..... nag_real_base (X02BHC)
- $e_{\max}$ ..... nag_real_max_exponent (X02BLC)
- $e_{\min}$ ..... nag_real_min_exponent (X02BKC)
- p ..... nag_real_base_digits (X02BJC)

4 Functions Withdrawn or Scheduled for Withdrawal

None.

5 References

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

ISO (1997) ISO Fortran 95 programming language (ISO/IEC 1539–1:1997)