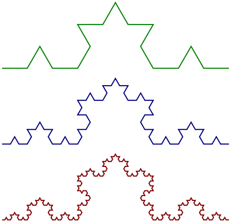
Aritmetica del Computador

Mg. Hermes Pantoja Carhuavilca

Universidad Nacional Mayor de San Marcos
Facultad de Ingenieria Industrial



Métodos Computacionales

Agenda

Introducción

Aritmetica del Computador

Sistema de Numeración

Representación de enteros

Base Binaria (2)

- 2 "**bits**" [0,1]
- 1011 en base $2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$ $= 8 + 0 + 2 + 1 = 11$ en base decimal

Sea N un número entero en base β tal que:

$$N = (a_n a_{n-1} a_{n-2} \dots a_2 a_1 a_0)_\beta = \sum_{k=0}^{n} a_k * \beta^k$$

Sistema de Numeración

Representación de números fraccionarios

$x = 0.7 = \frac{7}{10} = 7 \times 10^{-1}$

$x = 0.75 = 0.70 + 0.05 = 7 \times 10^{-1} + 5 \times 10^{-2}$

Sea x un número fraccionario en base β tal que:

$$x = (0.b_1 b_2 b_3 \ldots b_n)_\beta = b_1 \times \beta^{-1} + b_2 \times \beta^{-2} + \ldots + b_n \times \beta^{-n}$$

Base decimal (10)

- Potencia negativa de 10 para parte fraccionaria.
- $54.32 = 5 \times 10^1 + 4 * 10^0 + 3 \times 10^{-1} + 2 \times 10^{-2}$

Otros sistemas de numeración

- Mayor interes en decimal (10) y binario (2)
  Uso en computadores
- Otros sistemas
  octal (8), $\{0, 1, 2, ..., 7\}$
  hexadecimal (16), $\{0, 1, 2, ..., 9, A, B, C, D, E, F\}$

Conversión entre bases: ejemplos

Convertir de base 10 a cualquier base (s)

Ejemplo: convertir el número $(19)_{10}$ a binario.

$$
\begin{array}{l}
19 \,|\, 2 \\
a_0 \rightarrow 1 \quad 9 \,|\, 2 \\
a_1 \rightarrow 1 \quad 4 \,|\, 2 \\
a_2 \rightarrow 0 \quad 2 \,|\, 2 \\
a_3 \rightarrow 0 \quad 1 \leftarrow a_4
\end{array}
$$

$(19)_{10} = (10011)_2$

Ejemplo: convertir el número $(127)_{10}$ a hexadecimal.

$$
\begin{array}{l}
127 \,|\, 16 \\
a_0 \rightarrow 15 \quad 7 \,|\, 16 \\
a_1 \rightarrow 7 \quad 0 \leftarrow a_2
\end{array}
$$

$(127)_{10} = (07(15))_{16} = (07F)_{16}$

Conversión entre bases: ejemplos

Convertir de base 10 a cualquier base (s)

Ejemplo: $(0{,}1285)_{10}$ a base 4.

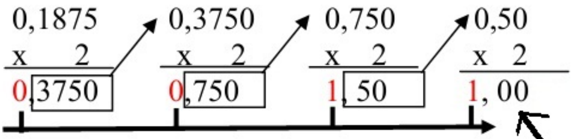
		$0{,}1285 \times 4$
a_{-1} →	0	$0{,}5140 \times 4$
a_{-2} →	2	$0{,}0560 \times 4$
a_{-3} →	0	$0{,}2240 \times 4$
a_{-4} →	0	$0{,}8960 \times 4$
a_{-5} →	3	$0{,}5840 \times 4$
a_{-6} →	2	$0{,}3360 \times 4$
	...	...

$(0{,}1285)_{10} = (0{,}020032\ldots)_4$

Ejemplo: $(0{,}3)_{10}$ a binario.

		$0{,}3 \times 2$
a_{-1} →	0	$0{,}6 \times 2$
a_{-2} →	1	$0{,}2 \times 2$
a_{-3} →	0	$0{,}4 \times 2$
a_{-4} →	0	$0{,}8 \times 2$
a_{-5} →	1	$0{,}6 \times 2$
a_{-6} →	1	$0{,}2 \times 2$
a_{-7} →	0	$0{,}4 \times 2$
a_{-8} →	0	$0{,}8 \times 2$
a_{-9} →	1	$0{,}6 \times 2$
	...	...

$(0{,}3)_{10} = (0{,}010011001\ldots)_2$



$$(0{,}1875)_{10} = (0{,}0011)_2$$

Parar cuando no existe
más parte fraccionaria.

Sistema de Punto Plotante

Definición (Sistema de Punto Flotante)

Un sistema de punto flotante se especifica por la base β, el largo de mantisa t, y límites para los exponentes de L, M. Un número de punto flotante tiene la forma

$$x = \pm 0.b_1 b_2 \ldots b_t \times \beta^e$$

donde $0.b_1 b_2 \ldots b_t$ es la mantisa, $b_1 \neq 0$ (para $x \neq 0$), $0 \leq b_i \leq \beta - 1$ para $2 \leq i \leq t$, y e el exponente el cual satisface $L \leq e \leq U$. El cero se representa con mantisa cero y exponente cero. El sistema de punto flotante se representa por

$$F(\beta, t, L, U)$$

Ejemplo

Tomando $(\beta, t, L, U) = (10, 2, -1, 2)$, *tenemos* 90 *posibles mantisas, y* 4 *exponentes, i.e.,* $-1, 0, 1, 2$. *Como hay dos posibles signos, tenemos un total de* $2(90)(4) + 1 = 721$ *números en el sistema.*

Nótese que el sistema de punto flotante es finito.
El Sistema de los números reales tiene a $\mathbb{R}$ como un conjunto inconmensurable porque no es posible representarlos a todos.
El Sistema de Punto Flotante es un subconjunto $\mathbb{F} \subset \mathbb{R}$ de números reales.

Definición (Cardinalidad)

Cardinalidad de $F(\beta, t, L, U)$:

$$2(\beta - 1)\beta^{t-1}(U - L + 1) + 1$$

Ejemplo

¿Cúantos números tendrá el sistema F(2,3,-1,2)?

Ejercicio

Dado el sistema de punto flotante $F(2,2,-1,2)$

1. $0.5 \in F$?
2. $3/4 \in F$?
3. $0.5 + 3/4 \in F$?

Puesto que la cantidad de números a almacenar es una cantidad finita, la mayoría de números reales tendrán que ser aproximados a aquellos que tienen una representación exacta en el sistema de punto flotante empleado. Esto origina las perdidas de precisión por redondeo.

Errores en el Computador

Los científicos que desarrollaron el cohete Ariane 5 vuelo 501 reutilizaron parte del código de su predecesor, el Ariane 4, pero los motores del cohete más nuevo incorporaron también, sin que nadie se diera cuenta, un “bug” en una rutina aritmética en la computadora de vuelo. Esto provocó, el 4 de junio de 1996, que la computadora fallara segundos después del despegue del cohete; 0,5 segundos más tarde falló el ordenador principal de la misión. El Ariane 5 se desintegró 40 segundos después del lanzamiento.

Aritmetica del Computador

Las operaciones de suma, resta, multiplicación y división en el sistema de punto flotante (F), se denota por $\oplus, \ominus, \otimes, \oslash$ respectivamente. Estas operaciones están definidas por:

$x \oplus y = fl(fl(x) + fl(y))$
$x \ominus y = fl(fl(x) - fl(y))$
$x \otimes y = fl(fl(x) \times fl(y))$
$x \oslash y = fl(fl(x) \div fl(y)), \quad fl(y) \neq 0, \;\; y \neq 0$

Estas operaciones no son cerradas sobre F, pues en algunos casos se genera underflow u overflow;

Ejemplo

Dado el sistema hipotético $F(10,3,-3,3)$ *y los números:*
$X = 2/30 = 0.066666\ldots$
$Y = 5/9 = 0.55555\ldots$
hallar $X \otimes Y$

Solución:
Valor Exacto de $X * Y = 10/270 = 0.037037037\ldots$
$fl(X) = 0.667 \times 10^{-1}$
$fl(Y) = 0.556 \times 10^{0}$
$fl(X) * fl(Y) = 0.667 \times 10^{-1} * 0.556 \times 10^{0} = 0.370852 \times 10^{-1}$
$x \otimes y = fl(fl(X) * fl(Y)) = 0.371 \times 10^{-1}$
Error$=10^{-4}$

Desbordamiento

Se puede producir cuando se operan dos datos y el resultado excede la capacidad de almacenamiento seleccionada.

Definición (Overflow)

Se produce cuando el número es muy grande y se excede el límite máximo de almacenamiento.

Definición (Underflow)

Se produce cuando el número es muy prqueño y se execede el límite mínimo de almacenamiento.

El Épsilon (ϵ) de la Máquina

Definición

El épsilon de la máquina es la distancia entre 1 *y el siguiente número máquina, se denota por eps.*

Ejercicio

Sea F el sistema de punto flotante caracterizado por $\beta = 2$,(base), $n = 4$(precisión), $m = -1$, $M = 2$, cada número en el conjunto F está representado por $\pm(.d_1 d_2 \ldots, d_n)_\beta \beta^e$ donde $m \leq e \leq M$

1. Cuál es el número más pequeño en valor absoluto del sistema F?
2. Demuestre que $3/4$ y $5/6$ pertenecen al sistema F, pero la suma "verdadera" de estos no pertenece a F.
3. Suponga que el tipo de error introducido en la representación de un número real en el sistema F es por redondeo. Como queda representado el numero $3/4 + 5/16$ en F. esto es:

$$\frac{3}{4} \oplus \frac{5}{16} = fl(\frac{3}{4} + \frac{5}{16}) =???$$

4. Encuentre el epsilon de la maquina.

Ejercicio

Recordar que "*fl*(*expresion*)" significa que todos los operandos son convertidos a números en punto flotante y todas las operaciones son desarolladas con la aritmética del punto flotante. Asuma $\beta = 10$, $t = 3$, $L = -3$, $U = 4$, y la aritmética es truncada. Obtener los valores de:

1. $fl(0.00009)$
2. $fl(3.146)$
3. $fl(9996)$
4. $fl((100.0 + 0.61) + 0.61)$ y $fl(100.0 + (0.61 + 0.61))$
5. $fl(2.34 \times (5.67 + 8.90))$ y
   $fl((2.34 \times 5.67) + (2.34 \times 8.90))$

Sistema de Punto Flotante

Es el conjunto de números de punto flotante normalizados, en el sistema de números con base β con t dígitos para la fracción (equivalentemente $t+1$ dígitos en la mantisa)

$$F(\beta, t, L, U) = \{a = \pm a_0.a_1 a_2 \ldots a_t \times \beta^e\}$$

$$\left\{ \left| \begin{array}{ll} a < a_0 < \beta & L \leq e \leq U \\ 0 \leq a_i < \beta & i = 1, 2, \ldots, t \end{array} \right. \right\} \cup \left\{ 0.\underbrace{00\ldots0}_{t \ veces} \right\}$$

La cantidad de números de punto flotante, que admite el sistema numérico $F(\beta, t, L, U)$ es
$2(\beta - 1)\beta^t(U - L + 1) + 1$

Representación de números del computador

Los computadores trabajan con aritmética real usando un sistema denominado de ”punto flotante”. Suponen un número real que tiene la expansión binaria:

Número Normalizado

Notación científica en base 2

Consiste en expresar los números no nulos en la forma:

$$\underbrace{\pm 1.d_1 d_2 \ldots. d_s \ldots.}_{\text{mantisa}} \bullet 2^{e}$$

exponente

con: $d_i \in \{0,1\}$ **y** $e \in \mathbb{Z}$ **(expresado también en base 2)**

Notación Normalizada

Aproximación de números reales en base 2 expresados en coma flotante normalizada

Técnicas de aproximación de mantisas de la forma:

$$\pm 1.d_1 d_2 \dots d_s d_{s+1} \dots$$

por mantisas con s dígitos decimales:

Truncado: $\pm 1.d_1 d_2 \dots d_s d_{s+1} \dots \approx \pm 1.d_1 d_2 \dots d_s$

Redondeo:

$$\pm 1.d_1 d_2 \dots d_s d_{s+1} \dots \approx \begin{cases} \pm 1.d_1 d_2 \dots d_s & \text{si } d_{s+1} = 0 \\ \left\{ \begin{array}{l} \pm(1.d_1 d_2 \dots d_s + \\ +0.0\ 0 \dots 1\ ) \end{array} \right\} & \text{si } d_{s+1} = 1 \end{cases}$$

"Suma binaria"

Ejemplo

Redondeo a mantisas con 6 dígitos binarios decimales:

$z = 1.0110111010....$

7º Dígito decimal

$$\begin{array}{rl} & \scriptstyle 0\,0\,0\,0\,1\,1 \\ & 1.011011 \\ + & 0.000001 \\ \hline & 1.011100 \end{array} = z^*$$

$z = 1.1111111010.....\ 2^e$

7º Dígito decimal

$$\begin{array}{rl} & \scriptstyle 1\,1\,1\,1\,1\,1\,1 \\ & 1.111111 \cdot 2^e \\ + & 0.000001 \cdot 2^e \\ \hline & 10.000000 \cdot 2^e \end{array}$$

Ajuste de exponentes: $1.000000 \cdot 2^{e+1} = z^*$

Almacenamiento de floats

Ejemplo

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Signo	Exponente (+/-)			Significando			

0 = +
1 = -

000 (especial)
001 (2^{-2})
010 (2^{-1})
011 (2^{0})
100 (2^{1})
101 (2^{2})
110 (2^{3})
111 (especial)

1.0000
1.0001
....
....
1.1111

1 = bit escondido

Almacenamiento de floats

- Mayor número positivo 0 110 1111 $= +2^3 \times 1.1111 = 2^3 \times (2 - 2^{-4}) = 1111.1 = 15.5$ decimal
- Menor número positivo 0 001 0000 $= +2^{-2} \times 1.0000 = 2^{-2} \times 2^0 = 0.01$ ó 0.25 decimal

Almacenamiento de floats

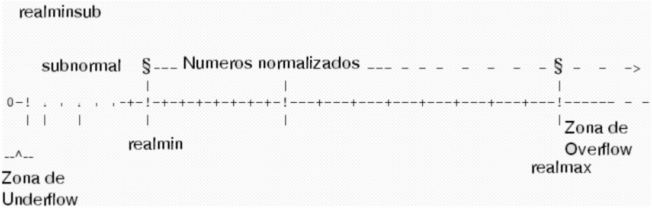
Combinaciones especiales de los exponentes:
000 Representación No Normalizada

- Mansitisa pasa a ser: 0.
- Exponente(000)=-2
- Menor número positivo pasa a ser $0\ 000\ 0001 = 2^{-2} \times 0.0001 = 2^{-2} \times 2^{-4} = 2^{-6} = 0.015625$

Además de las combinaciones especiales...
111 representación de infinito

- $01110000 = +$Infinito
- $11110000 = -$Infinito
- $11111000 =$Indeterminación
- Otras combinaciones $11111 - - - =$Not A Number (NaN)

Distribución de los datos en la recta numérica



Distribución de Números Puntos Flotante

Los números punto flotante no son uniformemente distribuidos sobre el eje real, como en el caso continuo. Veamos para el sistema $F(2, 2, -2, 1)$.

$(0.00)_2 \times 2^0 = 0$	$(1.01)_2 \times 2^{-1} = \left(1+\frac{1}{4}\right)\times\frac{1}{2} = \frac{5}{8}$	$(1.11)_2 \times 2^0 = \frac{7}{4}$
$(1.00)_2 \times 2^{-2} = 1 \times \frac{1}{4} = \frac{1}{4}$	$(1.10)_2 \times 2^{-1} = \left(1+\frac{1}{2}\right)\times\frac{1}{2} = \frac{3}{4}$	$(1.00)_2 \times 2^1 = 2$
$(1.01)_2 \times 2^{-2} = \left(1+\frac{1}{4}\right)\times\frac{1}{4} = \frac{5}{16}$	$(1.11)_2 \times 2^{-1} = \left(1+\frac{1}{2}+\frac{1}{4}\right)\times\frac{1}{2} = \frac{7}{8}$	$(1.10)_2 \times 2^1 = 3$
$(1.10)_2 \times 2^{-2} = \left(1+\frac{1}{2}\right)\times\frac{1}{4} = \frac{3}{8}$	$(1.00)_2 \times 2^0 = 1$	$(1.11)_2 \times 2^1 = \frac{7}{2}$
$(1.11)_2 \times 2^{-2} = \left(1+\frac{1}{2}+\frac{1}{4}\right)\times\frac{1}{4} = \frac{7}{16}$	$(1.01)_2 \times 2^0 = \frac{5}{4}$	$(1.01)_2 \times 2^1 = \frac{5}{2}$
$(1.00)_2 \times 2^{-1} = 1 \times \frac{1}{2} = \frac{1}{2}$	$(1.10)_2 \times 2^0 = \frac{3}{2}$	

Estándar IEEE-754

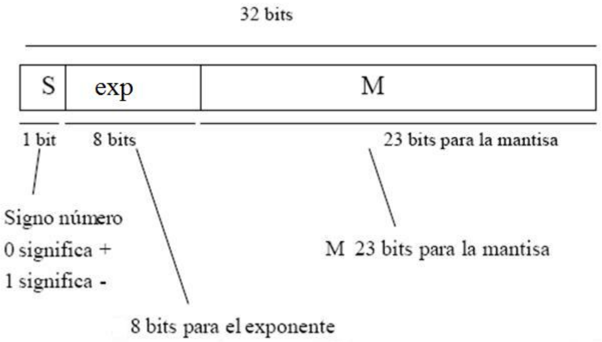
Este estándar se desarrolló para facilitar la portabilidad de los programas de un procesadora otro y para alentar el desarrollo de programas numéricos sofisticados. Este estándar ha sido ampliamente adoptado y se utiliza prácticamente en todos los procesadores y coprocesadores aritméticos actuales.

Estándar IEEE-754

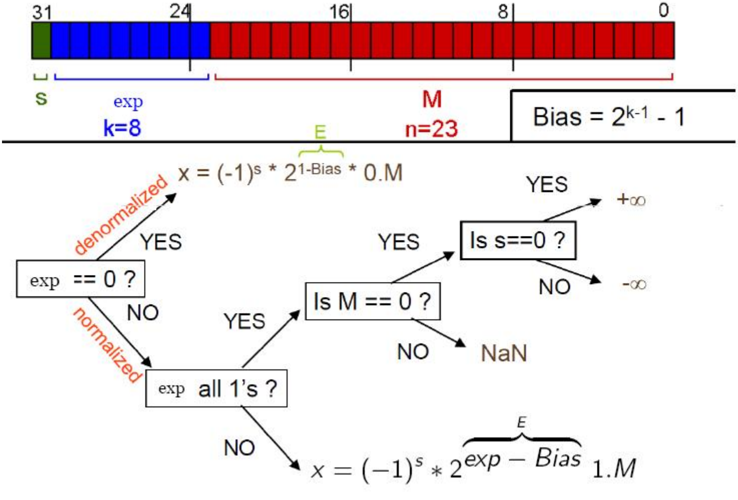
- El estándar del IEEE define el formato para precisión simple de 32 bits y para precisión doble de 64 bits.
- Hasta la década de los 90 cada computador utilizaba su propio formato en punto flotante, en 1985 se introduce el estándar IEEE-754 con la finalidad de uniformizarlos.

Estándar IEEE-754

Precisión Simple: 32 bits



Estándar IEEE-754



Ejemplos:

Represente en formato IEEE 754 simple precisión el número 3.25

Solución:

- Convertir a binario: $3.25 = 11.01$
- Normalizar el número (mover el punto decimal hasta que haya un solo 1 a la izquierda)
  $11.01 = 1.101 \times (2^1)$
- mantisa: 101
- exponente:
  $Bias = 2^{(8-1)} - 1 = 127$
  $exp = E + 127 \longrightarrow exp = 1 + 127 = 128 = 10000000$
- El número es positivo: bit de signo 0

Resultado: 01000000010100000000000000000000

Ejemplos:

Represente en formato IEEE 754 simple precisión el número 347.625

Solución:

- Convertir a binario: $347.625 = 101011011.101$
- Normalizar el número (mover el punto decimal hasta que haya un solo 1 a la izquierda)
  $101011011.101 = 1.01011011101 \times (2^8)$
- mantisa: 01011011101
- exponente:
  $Bias = 2^{(8-1)} - 1 = 127$
  $exp = E + 127 \longrightarrow exp = 8 + 127 = 135 = 10000111$
- El número es positivo: bit de signo 0

Resultado: 01000011101011011101000000000000

Ejemplos:

Ejemplo

¿Cuál es el valor de: 1 01111100 11000000000000000000000?

Solución:

- El bit de signo es 1:número negativo
- El exponente **exp** contiene $01111100 = 124$
- La mantisa es $0.11000\ldots = 0.75$

El valor es:

$$(-1)\times(1+0.75)\times(2^{(124-127)}) = -1.75\times(2^{(-3)}) = -0.21875$$

Notas importantes sobre el estándar IEEE 754

Como cero no es directamente representable en estándar IEEE 754, entonces dependiendo del exponente y la mantisa del número codificado, algunas representaciones tienen significados particulares, así como se resume en la siguiente tabla:

exp	M	Número
$0 < exp < E_{max}$	Cualquiera	$(-1)^s \times (1 + M) \times 2^{E-127}$
0	0	$Cero, (-1)^s \times 0$
0	$\neq 0$	$(-1)^s \times (0 + M) \times 2^{-126}$
E_{max}	0	Infinito, $(-1)^s \times \infty$
E_{max}	$\neq 0$	Not a Number, NaN

Números especiales Normalizados

El menor número positivo normalizado: $exp = 1, M = 0$

$$1.00000000000000000000000 \times 2^{1-127} = (1.0) \times 2^{-126}$$

El mayor número positivo normalizado:
$exp = 254, M = 11\ldots1$

$$1.11111111111111111111111 \times 2^{254-127}$$

Cualquier número mayor que este se dice **overflow**($exp = 255,\ M > 0$), el cual indica que el resultado es **Not a Number**, **NaN**

Números especiales SubNormalizados

El mayor número positivo subnormalizado:
$exp = 0, M = 11\ldots1$

$$0.11111111111111111111111 \times 2^{-126}$$

El menor número positivo subnormalizado:
$exp = 0, M = 00000000000000000000001$

$$0.00000000000000000000001 \times 2^{-126}$$

Cualquier número menor que este se dice **underflow** $(exp = 255,\ M > 0)$.

Estándar IEEE-754

Precisión Doble: 64 bits

64 bits

S	E	M
1 bit	11 bits	52 bits para la mantisa

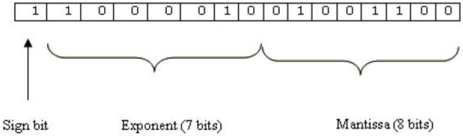
Signo número
0 significa +
1 significa -

11 bits para el exponente
Representacion exceso 1023

M 52 bits para la mantisa

Valor representado:
$\pm 1.M \times 2^{E-1023}$

Ejercicio:
Vamos a considerar un hipotético computador que en números de punto flotantes están representados en una palabra de 16-bit. Un ejemplo se muestra en la Figura 1:



Muestre la representación en punto flotante y los bits del:

1. El número eps (epsilón de la maquina)
2. Mayor valor positivo normalizado
3. Menor valor positivo normalizado
4. El número 1 y -10.375
5. El infinito y NaN

Solución

(a)

0	0	1	1	0	1	1	1	0	0	0	0	0	0	0	0

$(1.00000000)*2^{-8}$

(b)

0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1

$(1.11111111)*2^{63}$

(c)

0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

$(1.00000000)*2^{-62}$

Solución

(d) UNO

0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0

$(1.00000000)*2^0$

-10.375

1	1	0	0	0	0	1	0	0	1	0	0	1	1	0	0

$(-1)^1 \times (1.01001100)*2^3$

(e) INFINITO

0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

$((1.00000000)*2^{64}$

NAN

0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1

Ejercicio

Sea el sistema de punto flotante hipotético adecuado a la norma IEEE-754 que usará 16 bits con la siguiente estructura:

Signo (s=1 bit)	Exponente (k=5 bits)	Mantisa (t=10 bits)

Muestre como se almacena en binario:

1. El epsilon de la maquina
2. El mayor número positivo no normalizado
3. El menor número positivo no normalizado
4. El número -43.000001
5. El -0
6. El -Inf

Bibliografía

Richard L. Burden and J. Douglas Faires
Análisis numérico, 7a ed.

Steven C. Chapra and Raymond P. Canale
Métodos numéricos para ingenieros, 5a ed.