TERMINOLOGÍA ASOCIADA CON LA REPRESENTACIÓN DE NÚMEROS EN EL COMPUTADOR

bit Hace referencia a un dígito binario (0 ó 1) y proviene de la contracción de las palabras inglesas "***b****inary dig****it***". Es la unidad de almacenamiento en el computador y está asociada a una celda electrónica.

byte Es un conjunto de 8 bits. Así, en Matlab, un número ocupa 8 bytes, lo cual equivale a 64 bits. Esto se debe a que Matlab trabaja automáticamente en precisión doble.

palabra En el computador, una *palabra* es un conjunto de bits o celdas electrónicas, que representan un número en base 2. En la mayoría de las arquitecturas computacionales actuales, las palabras del computador ocupan 32 bits (existen, sin embargo, arquitecturas de 64 bits para PC en la actualidad).

precisión simple En el estándar IEEE 754, se refiere al almacenamiento de un número usando de 32 bits, siguiendo dicho estándar .

precisión doble En el estándar IEEE 754, se refiere al almacenamiento de un número usando 64 bits, siguiendo dicho estándar .

número de máquina Se refiere a cualquier número que se obtenga a partir de la representación de punto flotante normalizada o desnormalizada bajo el estándar IEEE 754. Se incluye al cero y a los números enteros representables.

TERMINOLOGÍA ASOCIADA CON LA REPRESENTACIÓN DE NÚMEROS EN EL COMPUTADOR

redondeo Es el proceso de aproximar un número no de máquina por un número de máquina. Más específicamente, si el número de máquina que se elige para la aproximación es aquel que está más cerca del número dado, se habla de *redondeo correcto*.

error de redondeo Es el error que se comete al aproximar un número que no es de máquina por un número de máquina. Resulta que el error de redondeo (relativo) está acotado por el épsilon de la máquina.

épsilon de la máquina Es el primer número de máquina que es mayor que 1. De acuerdo al estándar IEEE 754, el épsilon de la máquina en precisión simple es $2^{-23} \approx 1.1920929e\text{-}007$, mientras que en precisión doble es $2^{-52} \approx 2.220446049250313e\text{-}016$.
En Matlab estas cantidades son proporcionadas por la función **eps**, invocada con el parámetro *'single'* en el primer caso y sin parámetros en el segundo.
En general, si en la representación de punto flotante se usan **n** bits para almacenar la mantisa, entonces el épsilon de la máquina es 2^{-n}.

TERMINOLOGÍA ASOCIADA CON LA REPRESENTACIÓN DE NÚMEROS EN EL COMPUTADOR

épsilon de la máquina El épsilon de la máquina posee una caracterización que suele ser útil para calcularlo sin conocer el estándar de almacenamiento del computador. Dicha caracterización es la siguiente:
"El épsilon de la máquina es el número más pequeño de la forma 2^{-k} *(k entero positivo) tal que* $1 + 2^{-k} \neq 1$."
Quiere decir que en el computador alguna potencia negativa de 2 no le agrega nada al número 1, es decir, debe existir un entero positivo m tal que $1+2^{-m} = 1$. ¿Cuál es el valor más pequeño de m en precisión simple y doble?

El conjunto de números de máquina posee un conjunto de características importantes, a saber:

- Es un conjunto finito y discreto.
- Es simétrico respecto al cero (existe la misma cantidad de números de máquina negativos y positivos).
- Los números de máquina se aglomeran a medida que se acercan al cero.

TERMINOLOGÍA ASOCIADA CON LA REPRESENTACIÓN DE NÚMEROS EN EL COMPUTADOR

Existe un número de máquina más grande:
en precisión simple es

$$2^{127} \times (1.111\ldots11)_2 = 2^{127} \times (1 + \sum_{k=1}^{23} 2^{-k}) \approx 3.4028235e+038$$

23 unos después del punto

en precisión doble es

$$2^{1023} \times (1.111\ldots11)_2 = 2^{1023} \times (1 + \sum_{k=1}^{52} 2^{-k}) \approx 1.7976931348623 16e+308$$

52 unos después del punto

Estos valores son suministrados por la función ***realmax*** de MATLAB, pasándole el argumento '***single***' y ningún argumento para precisión doble, esto es, ***realmax('single')*** en el primer caso y ***realmax*** en el segundo caso.

TERMINOLOGÍA ASOCIADA CON LA REPRESENTACIÓN DE NÚMEROS EN EL COMPUTADOR

Existe un número de máquina positivo normalizado más pequeño:
en precisión simple es

$$2^{-126} \times (1.000\ldots00)_2 = 2^{-126} \times 1 \approx 1.1754944e-038$$

23 ceros después del punto

en precisión doble es

$$2^{-1022} \times (1.000\ldots00)_2 = 2^{-1022} \times 1 \approx 2.2250738585072 01e-308$$

52 ceros después del punto

Estos valores son suministrados por la función ***realmin*** de MATLAB, pasándole el argumento '***single***' y ningún argumento para precisión doble, esto es, ***realmin('single')*** en el primer caso y ***realmin*** en el segundo caso.

TERMINOLOGÍA ASOCIADA CON LA REPRESENTACIÓN DE NÚMEROS EN EL COMPUTADOR

Existe un número de máquina positivo **no normalizado** más pequeño:
en precisión simple es

$$2^{-126} \times (0.000\ldots01)_2 = 2^{-126} \times 2^{-23} = 2^{-149} \approx 1.4012985e-045$$

23 dígitos después del punto

en precisión doble es

$$2^{-1022} \times (0.000\ldots01)_2 = 2^{-1022} \times 2^{-52} \approx 4.9406564584124654e-324$$

52 dígitos después del punto

De acuerdo a lo dicho en las láminas anteriores, se puede asegurar que estos valores pueden ser obtenidos en Matlab de la siguiente manera:

en precisión simple es **realmin**(*'single'*) * **eps**(*'single'*)

en precisión doble es **realmin** * **eps**

TERMINOLOGÍA ASOCIADA CON LA REPRESENTACIÓN DE NÚMEROS EN EL COMPUTADOR

La existencia de los números de máquina no normalizados evita que exista un "salto" grande entre el número de máquina positivo normalizado más pequeño y el cero, permitiendo una progresión "suave" de los números de máquina hacia el cero.

A medida que se usen más bits para representar a los números de punto flotante, y, por ende, se represente la mantisa con más bits también, menos será necesario disponer de estos números de máquina no normalizados. Sin embargo, ya forman parte del estándar y lo más seguro es que permanezcan allí por mucho tiempo.

Una consecuencia de la existencia de los números de máquina no normalizados es que la comparación entre un número cualquiera y 0 usando el operador == en Matlab resulta muy severa, en el sentido de que sólo será cierta cuando el orden de magnitud del número sea inferior a 10^{-324}. Esta es la fuente de muchos errores de programación usando Matlab.

TERMINOLOGÍA ASOCIADA CON LA REPRESENTACIÓN DE NÚMEROS EN EL COMPUTADOR

En relación a la representación de los números de punto flotante bajo el estándar IEEE 754, se puede decir que:

La mantisa está relacionada con la **precisión** o **exactitud** de la representación de los números de punto flotante.
Esto se debe a que es en la mantisa donde se almacenan los dígitos significativos de los números.

El exponente tiene que ver con el **tamaño** u **orden de magnitud** de los números de punto flotante representados.

Así, se obtendrá una representación más precisa de los números en el computador cuando se aumente la cantidad de bits para almacenar la mantisa, y se obtendrán números en un rango mayor de tamaños cuando se aumente la cantidad de bits para almacenar el exponente.

Cuando se usan más bits en general para representar los números en el computador, aumentan tanto la cantidad de bits usados para la mantisa como para el exponente (tal como se vio en el estándar IEEE 754), por lo tanto al cambiarse a una arquitectura que use más bits en general, estaremos mejorando simultáneamente la precisión en la representación así como el rango de tamaños de los números representados.

TERMINOLOGÍA ASOCIADA CON LA REPRESENTACIÓN DE NÚMEROS EN EL COMPUTADOR

A fin de interpretar más fácilmente el significado de la precisión en la representación de los números en el computador, volvamos a los valores del épsilon de la máquina.

En el caso de precisión simple, sabemos que dicho valor es $2^{-23} \approx 1.1920929e\text{-}007$, es decir, es del orden de 10^{-7}. Por lo tanto, los 23 bits usados para representar la mantisa (dígitos binarios significativos) corresponden aproximadamente a 7 dígitos decimales. Esto quiere decir que **en precisión simple (32 bits) no se puede esperar tener más de 7 dígitos significativos exactos**.

Por otro lado, en precisión doble, el épsilon de la máquina es $2^{-52} \approx 2.2204460493e\text{-}016$, es decir, del orden de 10^{-16}. Por lo tanto, los 52 bits usados para representar la mantisa (dígitos binarios significativos) corresponden aproximadamente a 16 dígitos decimales. Esto quiere decir que **en precisión doble (64 bits) no se puede esperar tener más de 16 dígitos significativos exactos**.

De hecho, estas cifras son límite, de modo que no es conveniente aspirar a alcanzarlas exactamente. Por ejemplo, aspirar a 5 y 14 cifras decimales signifiactivas en precisión simple y doble, respectivamente, es razonable. Por supuesto, esto depende del problema particular que se esté resolviendo.

TERMINOLOGÍA ASOCIADA CON LA REPRESENTACIÓN DE NÚMEROS EN EL COMPUTADOR

Observación:

Al realizar una operación cualquiera en el computador, aun cuando los números involucrados sean números de máquina de punto flotante, el resultado, en general, ***no*** es un número de máquina. Es por ello que se dice que el error de redondeo siempre ocurre y es inevitable, en términos generales, cuando se efectúan operaciones entre números de punto flotante en el computador. Lo que importa es estimar cuánto se ve afectado el resultado final de una secuencia de cálculos por dichos errores de redondeo.

TERMINOLOGÍA ASOCIADA CON LA REPRESENTACIÓN DE NÚMEROS EN EL COMPUTADOR

La **representación de números enteros** en el computador bajo el estándar IEEE es mucho más simple que la representación de números de punto flotante, ya que en el caso de los enteros sólo se requiere almacenar el signo del número y sus dígitos.

En lugar de dedicar un bit para el signo del número entero, el estándar hace una consideración similar a la hecha para el exponente de la representación de punto flotante. Es decir, se usan todos los bits para representar un cierto rango de números enteros y se resta un sesgo para centrar dicho rango en cero.

Por ejemplo, en precisión simple (32 bits) se genera el rango de valores que va desde 0 (los 32 bits todos nulos) hasta $2^{32} - 1 = 4294967295$ (los 32 bits todos iguales a 1). El sesgo en este caso es 2147483648, de modo que al restarlo se obtiene el rango final de números enteros en precisión simple, que va desde - 2147483648 hasta 2147483647. En Matlab, estos valores son suministrados, respectivamente, por las funciones **intmin** e **intmax**, invocadas sin parámetros.

En doble precisión la idea es similar, y el rango de números enteros resultante va desde -9223372036854775808 hasta 9223372036854775807. En Matlab se obtienen estos valores invocando las mismas funciones mencionadas en el párrafo anterior, pero suministrándoles el parámetro *'int64'*, esto es, se invocan **intmin**(*'int64'*) e **intmax** (*'int64'*). ¿Cuál es el sesgo para la representación de enteros en el caso de precisión doble? ¡Es muy fácil obtenerlo!