Computación
para Estudiantes de Física

Apunte Número 1

Datos Numéricos:
Representación y Almacenamiento

Sergio J. Sciutto
Departamento de Física
Facultad de Ciencias Exactas
Universidad Nacional de La Plata
C. C. 67 - 1900 La Plata, Argentina.

AÑO 2010

1. Representación de números reales

Cada uno de los números enteros $b > 1$ puede ser utilizado como base para representar números reales. En efecto, puede demostrarse que para todo número real $r \in \mathbb{R}$ existe uno y solo un conjunto de números enteros c_i, $0 \leq c_i < b$, $-\infty < i < M-1$ (M entero), $c_{M-1} \neq 0$ si $r \neq 0$, tal que la serie definida en la siguiente ecuación converge a r:

$$r = s \sum_{i=-\infty}^{M-1} c_i b^i \tag{1}$$

El límite máximo, $M-1$, es siempre finito, y $s = \pm 1$ es el *signo* de r ($s = 1$ si $r \geq 0$, -1 si $r < 0$).

Cada uno de los coeficientes c_i se denomina *dígito* (en base b) y la serie suele notarse de la siguiente manera

$$r = s\,(c_{M-1}\,c_{M-2}\,\ldots\,c_1\,c_0\,,c_{-1}\,c_{-2}\,\ldots)_b, \tag{2}$$

en donde el subíndice b indica explícitamente la base utilizada.

Ejemplo. Sea $r = 17/3 = 5{,}666666\ldots$ Se verifica: $b = 10$, $s = 1$, $M = 1$, $c_0 = 5$, $c_{-1} = c_{-2} = \cdots = 6$.

Si todos los coeficientes c_{-j} ($j > 0$) son nulos, el número r es entero, y la serie se reduce a un desarrollo de M dígitos en base b:

$$r = s\,(c_{M-1}\,c_{M-2}\,\ldots\,c_1\,c_0)_b. \tag{3}$$

Si alguno de los coeficientes c_{-j} es no nulo, el número no es entero. Si $c_{-j} = 0$ para todo $j > j_0 > 0$ o si $c_j = f(c_{-1}, c_{-2}, \ldots, c_{-j+1})$ para todo $j > j_0' > 0$ (recurrencia), entonces el número r pertenece al conjunto de los números racionales. Si la sucesión c_{-j} no termina y no es recurrente, entonces el número al cual converge es irracional.

Para todo número real r, el desarrollo (1) puede colocarse en la forma

$$r = s\,b^M \sum_{j=1}^{\infty} c_{M-j} b^{-j} = s\,b^M \sum_{j=1}^{\infty} d_j b^{-j} \tag{4}$$

o, utilizando notación compacta

$$r = s\,b^M\,(0\,,d_1\,d_2\,d_3\,\ldots)_b. \tag{5}$$

Esta representación del número r se denomina representación de r en *punto flotante.* Toda vez que $d_1 \neq 0$ la representación se dice *normalizada.* La constante M se denomina *característica* del número y la sucesión $d_1\,d_2\,d_3\,\ldots$ es su *mantisa.* Tanto la característica como la mantisa dependen de la base elegida.

2. Cambio de base

Sean $b_1 > 1$ y $b_2 > 1$ dos enteros. Sea

$$r = s\,(c_{M-1}\,c_{M-2}\,\ldots\,c_1\,c_0\,,c_{-1}\,c_{-2}\,\ldots)_{b_1}. \tag{6}$$

Se desea hallar la representación del número r en la base b_2:

$$r = s'\,(c'_{M'-1}\,c'_{M'-2}\,\ldots\,c'_1\,c'_0\,,c'_{-1}\,c'_{-2}\,\ldots)_{b_2}, \tag{7}$$

en donde $0 \leq c'_i < b_2$ y $c'_{M'-1} \neq 0$ si $r \neq 0$.

Claramente, $s' = s$, es decir, el signo del número no depende de la base b en la que se representa al mismo.

Para obtener los otros coeficientes se procede de la siguiente manera:

Parte entera. Sea $\alpha = [|r|]$, es decir[1], $\alpha = s(c_{M-1}\ldots c_0)_{b_1}$. Los coeficientes (dígitos) de la representación en base b_2 se calculan realizando la siguiente serie de divisiones enteras (aritmética en base b_1):

$$\begin{array}{lcl} \alpha_0 & = & \alpha \\ \alpha_0 & = & \alpha_1 b_2 + \tilde{c}_0 \\ \alpha_1 & = & \alpha_2 b_2 + \tilde{c}_1 \\ \vdots & & \vdots \\ \alpha_{k-1} & = & \alpha_k b_k + \tilde{c}_{k-1} \\ \vdots & & \vdots \\ \alpha_{M'-1} & = & \alpha_{M'} b_k + \tilde{c}_{M'-1} \end{array} \tag{8}$$

Esta serie de operaciones es finita; finaliza cuando $\alpha_{M'} = 0$, y esta condición define el valor de la característica M'. Los dígitos c'_i, $0 \leq i \leq M'-1$, son las representaciones en base b_2 de los correspondientes restos $\tilde{c}_i$, los cuales verifican $0 \leq \tilde{c}_i < b_2$, $\tilde{c}_{M'-1} \neq 0$.

Parte fraccionaria. Sea $\beta = |r| - \alpha = (0\,,c_{-1}\,c_{-2}\,\ldots)_{b_1}$ la mantisa de $|r|$. Los coeficientes (dígitos) en base b_2 pueden obtenerse realizando la siguiente serie de multiplicaciones (aritmética en base b_1):

$$\begin{array}{lcllcllcl} \beta_0 & = & \beta & & & & & & \\ \beta'_1 & = & \beta_0 b_2, & \tilde{c}_{-1} & = & [\beta'_1], & \beta_1 & = & \beta'_1 - \tilde{c}_{-1} \\ \beta'_2 & = & \beta_1 b_2, & \tilde{c}_{-2} & = & [\beta'_2], & \beta_2 & = & \beta'_2 - \tilde{c}_{-2} \\ \vdots & & \vdots & \vdots & & \vdots & \vdots & & \vdots \\ \beta'_j & = & \beta_{j-1} b_2, & \tilde{c}_{-j} & = & [\beta'_j], & \beta_j & = & \beta'_j - \tilde{c}_{-j} \\ \vdots & & \vdots & \vdots & & \vdots & \vdots & & \vdots \end{array} \tag{9}$$

[1]El símbolo $[x]$ indica la parte entera de x.

Este proceso es finito sólo si $\beta_j = 0$ para algún j, en caso contrario la sucesión de coeficientes es infinita. Los dígitos c_{-j}, $j > 0$ son las representaciones en base b_2 de las correspondientes partes enteras $\tilde{c}_{-j}$, que verifican $0 \leq \tilde{c}_{-j} < b_2$.

Cuando la parte fraccionaria de una o ambas sucesiones es infinita, puede ser necesario tener que adoptar un criterio de truncamiento de las series. Si se considera una parte fraccionaria de k_1 dígitos en la representación en base b_1, la magnitud de los dígitos no considerados es, a lo sumo, $b_1^{-k_1}$. Si se desea representar en base b_2 al mismo número con similar precisión, entonces se deberá tomar k_2 dígitos, de modo que $b_2^{-k_2} \leq b_1^{-k_1}$, es decir

$$k_2 \geq k_1 \frac{\log b_1}{\log b_2}. \tag{10}$$

Es evidente que el número de dígitos k_2 es proporcional a k_1. La constante de proporcionalidad depende de las bases b_1 y b_2.

3. Error en la representación de un número

Sea r un número real. Existen muchos casos en donde no resulta posible operar directamente con el *valor exacto* de dicho número, debiéndose en cambio utilizar un *valor aproximado.* Cuando sea necesario distinguir claramente entre ambos valores, lo haremos utilizando la notación $\overline{r}$ y r^* para los valores exacto y aproximado de r, respectivamente.

Sea entonces r^* una aproximación a r. Se define el *error absoluto* cometido al representar r con r^* como

$$\epsilon_{\text{abs}} = |r - r^*|. \tag{11}$$

Si $r \neq 0$ se define el *error relativo* de la aproximación como

$$\epsilon_{\text{rel}} = \left| \frac{r - r^*}{r} \right|. \tag{12}$$

Cuando r^* contiene una representación de r con un número finito de dígitos, estos errores serán, en general, no nulos.

3.1. Redondeo y truncamiento

Sea $r = s(c_{M-1} \ldots c_0 \, , c_{-1} \ldots c_{-k} \ldots)_b$. Se puede definir la representación de r con k dígitos fraccionarios de dos modos diferentes:

Truncamiento. Se toman los primeros k dígitos y los demás se ignoran:

$$r^* = s \, (c_{M-1} \ldots c_0 \, , c_{-1} \ldots c_{-k})_b \tag{13}$$

El error absoluto resulta, entonces:

$$\epsilon_{\text{abs}} = |(0, \overbrace{0 \ldots 0}^{k} \, c_{-k-1} \, c_{-k-2} \ldots)_b| \tag{14}$$

$$= \left| b^{-k-1} \sum_{j=0}^{\infty} c_{-k-1-j} b^{-j} \right| < b^{-k}. \tag{15}$$

Es decir,

$$\epsilon_{\text{abs}} < b^{-k} \tag{16}$$

(el error es menor que lo que representa un dígito en la última posición considerada). Es de notar que los errores de truncamiento son siempre de un mismo signo.

Redondeo. Se analiza la "cola" a partir de la posición k:

$$t = b^{-k-1} \sum_{j=0}^{\infty} c_{-k-1-j} b^{-j}. \tag{17}$$

Si $t < b^{-k}/2$ entonces r^* se elige igual que en el caso de truncamiento. Si $t \geq b^{-k}/2$ entonces se toma

$$r^* = s \, (c_{M-1} \ldots c_0 \, , c_{-1} \ldots c_{-k})_b + b^{-k} \tag{18}$$

$$= s \, (c'_{M-1} \ldots c'_0 \, , c'_{-1} \ldots c'_{-k})_b \tag{19}$$

El error absoluto esta dado por

$$\epsilon_{\text{abs}} \leq \begin{cases} |t| & \text{para } t < b^{-k}/2 \\ |b^{-k} - t| & \text{para } t \geq b^{-k}/2, \end{cases} \tag{20}$$

es decir,

$$\epsilon_{\text{abs}} \leq \frac{b^{-k}}{2}. \tag{21}$$

El signo del error de redondeo es variable.

En el caso de números flotantes normalizados, tomando a k como el número de dígitos en la mantisa, es decir,

$$r^* = s \, b^M \, (0{,}d'_1 \, d'_2 \, \ldots d'_k)_b, \tag{22}$$

el error *relativo* de la aproximación vale

$$\epsilon_{\text{rel}} \leq \begin{cases} b^{-k} & \text{(truncamiento)} \\ b^{-k}/2 & \text{(redondeo)} \end{cases} \tag{23}$$

4. Propagación de errores en operaciones aritméticas de precisión finita

4.1. Propagación de errores en las operaciones fundamentales

Suma y resta. Sean x e y reales, conocidos con incertezas absolutas δx y δy, respectivamente. Sea $S = x + y$, y sea $\overline{S}$ el valor exacto de S. El error absoluto de S es

$$\delta S = |S - \overline{S}| = |(x + y) - (\overline{x} + \overline{y})| = |(x - \overline{x}) + (y - \overline{y})| \leq |x - \overline{x}| + |y - \overline{y}|, \quad (24)$$

es decir,

$$\delta S \leq \delta x + \delta y, \quad (25)$$

resultado que nos indica que el error absoluto de la suma es menor o igual que la suma de los errores absolutos de los sumandos.

Cuando $S \neq 0$ es posible calcular el error relativo, dado por

$$\delta_r S = \frac{\delta x + \delta y}{|S|}. \quad (26)$$

Con un poco más de generalidad, dados a y b factores conocidos exactamente, si $S = ax + by$, entonces es inmediato demostrar que

$$\delta S \leq |a|\, \delta x + |b|\, \delta y. \quad (27)$$

Observesé que los coeficientes deben ser tomados en *valor absoluto.* Este detalle adquiere especial relevancia en el caso de la resta ($a = 1$, $b = -1$) en donde el error absoluto resulta igual al de la suma. En aquellos casos en donde $0 \simeq |x - y| \ll |x + y|$ el error relativo (26) puede llegr a ser muy grande.

Multiplicación y división. Sea $P = x^\alpha y^\beta$ con α y β reales conocidos exactamente (eventualmente iguales a 1), y x e y no nulos. Asumiendo que los errores relativos $\delta_r x = \delta x/|x|$ y $\delta_r y = \delta y/|y|$ son pequeños, el siguiente desarrollo en serie resulta justificado:

$$\overline{P} + \delta P = \overline{P} + \frac{\partial P}{\partial x}\, \delta x + \frac{\partial P}{\partial y}\, \delta y. \quad (28)$$

Entonces,

$$\delta P \leq \left|\frac{\partial P}{\partial x}\right| \delta x + \left|\frac{\partial P}{\partial y}\right| \delta y, \quad (29)$$

es decir,

$$\delta P \leq |\alpha x^{\alpha-1} y^\beta|\, \delta x + |\beta x^\alpha y^{\beta-1}|\, \delta y; \quad (30)$$

de donde el error relativo $\delta_r P = \delta P/|P| = |x^{-\alpha} y^{-\beta}| \delta P$ puede expresarse como:

$$\delta_r P = |\alpha|\, \delta_r x + |\beta|\, \delta_r y. \tag{31}$$

Esta expresión nos indica que el error relativo de productos P es igual a la suma de los errores relativos de los factores, multiplicados por los respectivos exponentes. En el caso particular de la mutiplicación y la division es $|\alpha| = |\beta| = 1$, y entonces $\delta_r P = \delta_r x + \delta_r y$.

4.2. Propagación de errores en cadenas de operaciones aritméticas de punto flotante.

Consideremos ahora que los números x e y de los párrafos precedentes son expresados en punto flotante con un número finito de dígitos en sus mantisas, el cual esta fijado de antemano. Esto significa que ambos números son conocidos con un cierto error relativo máximo ϵ_r, común a ambos números (ver sección 3.1).

Consideremos en primer término la multiplicación. Si $P = xy$ entonces el error absoluto del producto, ecuación (31), se puede escribir como

$$\delta P \leq 2|xy|\, \epsilon_r. \tag{32}$$

Al realizar un producto entre dos números expresados en punto flotante, puede resultar necesario relimitar (truncar o redondear) y/o renormalizar el resultado, tal como se ilustra en el siguiente ejemplo: Sean $x = 0{,}12345 \times 10^{-5}$ e $y = 0{,}31001 \times 10^{9}$ dos números representados en punto flotante normalizado con 5 dígitos decimales ($\epsilon_r = 5 \times 10^{-5}$). Se verifica

$$\begin{array}{lcll} P & = & 0{,}12345 \times 0{,}31001 \times 10^{-5+9} & \\ P & = & 0{,}0382707345 \times 10^{4} & \text{producto} \\ P & = & 10^{3} \times 0{,}38271 & \text{redondeo y normalización} \end{array} \tag{33}$$

El error absoluto del producto es $\delta P \simeq 4 \times 10^{-2}$.

Para sumar números expresados en punto flotante es necesario en primer término realizar la operación de *alineado* de los números, que consiste en "alinear" las mantisas de números con diferentes características, a fin de poder proceder con la suma "columna a columna". Más explicitamente: Sean

$$x = s_x b^{M_x} \sum_{j=1}^{k} d_j^x b^{-j} \qquad y = s_y b^{M_y} \sum_{j=1}^{k} d_j^y b^{-j} \tag{34}$$

los desarrollos de x e y en punto flotante normalizado con k dígitos; y sea $M = \text{máx}(M_x, M_y)$; entonces estos desarrollos pueden escribirse como

$$x = s_x b^{M} \sum_{j=1}^{k} d_j^x b^{-j-(M-M_x)} \qquad y = s_y b^{M} \sum_{j=1}^{k} d_j^y b^{-j-(M-M_x)}. \tag{35}$$

Estos nuevos desarrollos tienen la misma característica (M), pero uno de ellos es no normalizado (el exponente de b para $j = 1$ no es -1). Por poseer igual característica, se dice que las mantisas han sido "alineadas" y se puede proceder a realizar su suma $S = x + y$. El resultado debe ser luego limitado (truncado o redondeado según corresponda) a k dígitos, y normalizado si corresponde.

Consideremos por ejemplo la suma de los siguientes números expresados en punto flotante normalizado con 5 dígitos decimales: $x = 0{,}99876 \times 10^3$, $y = 0{,}72041 \times 10^1$. Se verifica $M = \text{máx}(3, 1) = 3$, luego

$$\begin{array}{rcll} x & = & 10^3 \times 0{,}99876 & \\ y & = & 10^3 \times 0{,}0072041 & \text{alineación} \\ S & = & 10^3 \times 1{,}0059641 & \text{suma alineada} \\ S & = & 10^4 \times 0{,}10060 & \text{redondeo y normalización} \end{array} \tag{36}$$

es decir, el resultado de la suma expresado en punto flotante normalizado de 5 dígitos decimales es $x + y = 0{,}10060 \times 10^4$. Notar que en este caso la suma alineada se ha realizado en forma "exacta", es decir, considerando todos los dígitos no nulos de ambos sumandos. En otros ejemplos podría ser posible tener que considerar además limitaciones en el número de dígitos a utilizar al momento de realizar la suma alineada.

El proceso de alineación modifica el error que afecta al sumando alineado. Es evidente que el error absoluto de éste término al efectuar la suma y renormalizar el resultado resulta similar al error del término de máxima característica (que no requiere ser alineado), entonces la ecuación (25) puede escribirse

$$\delta S = 2\,\text{máx}(\delta x, \delta y) = 2\,\text{máx}(|x|, |y|)\,\epsilon_r. \tag{37}$$

Cuando las operaciones a realizar son numerosas, los errores de truncamiento se van amplificando, y pueden degradar significativamente la calidad del resultado final.

Consideremos el siguiente ejemplo ilustrativo: Se desea sumar n números: $S = \sum_{i=1}^{n} x_i$. Esta operación se realiza en la práctica a través de una sucesión de sumas individuales:

$$\begin{array}{rcl} S_1 & = & x_1 \\ S_2 & = & S_1 + x_2 \\ S_3 & = & S_2 + x_3 \\ \vdots & \vdots & \vdots \\ S = S_n & = & S_{n-1} + x_n. \end{array} \tag{38}$$

Aplicando reiteradamente (37) se tiene

$$\begin{array}{rcl} \delta S_1 & = & |x_1|\,\epsilon_r \\ \delta S_2 & = & 2\,\text{máx}(|S_1| + |x_2|)\,\epsilon_r \\ \delta S_3 & = & 2\,\text{máx}(|S_2| + |x_3|)\,\epsilon_r \\ \vdots & \vdots & \vdots \\ \delta S = \delta S_n & = & 2\,\text{máx}(|S_{n-1}| + |x_n|)\,\epsilon_r. \end{array} \tag{39}$$

El error de la suma S_i es proporcional al máximo valor registrado en las cantidades consideradas hasta ese momento. Por este motivo, al sumar una larga sucesión de números resulta conveniente comenzar por los de menor valor absoluto, de modo de mantener las sumas parciales (y los errores absolutos) tan reducidos como sea posible. En algunas circunstancias, la realización de sumas en el orden inadecuado puede conducir a resultados incorrectos. Esta es una característica de la aritmética a precisión finita, que no se presenta en las operaciones exactas en donde la suma es conmutativa.

Consideremos por ejemplo la suma $S = \sum_{i=1}^{100} x_i$, con

$$x_i = 1 + \cos\left(\frac{\pi}{180}\right) \tanh[2(3-i)] \tag{40}$$

utilizando aritmética flotante de 4 dígitos. Con esta precisión, los términos x_i resultan:

i	x_i
1	$0{,}1999 \times 10^{1}$
2	$0{,}1964 \times 10^{1}$
3	$0{,}1000 \times 10^{1}$
4	$0{,}3612 \times 10^{-1}$
5	$0{,}8229 \times 10^{-3}$
6	$0{,}1646 \times 10^{-3}$
7	$0{,}1525 \times 10^{-3}$
$\vdots$	$\vdots$
99	$0{,}1523 \times 10^{-3}$
100	$0{,}1523 \times 10^{-3}$

Sumando de izquierda a derecha se obtiene:

$$\begin{aligned}
S_2 = x_1 + x_2 &= 10^1 \times 0{,}1999 + 10^1 \times 0{,}1964 = 10^1 \times 0{,}3963 \\
S_3 = S_2 + x_3 &= 10^1 \times 0{,}3963 + 10^1 \times 0{,}1000 = 10^1 \times 0{,}4963 \\
S_4 = S_3 + x_4 &= 10^1 \times 0{,}4963 + 10^{-1} \times 0{,}3612 = 10^1 \times 0{,}4999 \\
S_5 = S_4 + x_5 &= 10^1 \times 0{,}4999 + 10^{-3} \times 0{,}8229 = 10^1 \times 0{,}5000 \\
S_6 = S_5 + x_6 &= 10^1 \times 0{,}5000 + 10^{-3} \times 0{,}1646 = 10^1 \times 0{,}5000 \\
S_7 = S_6 + x_7 &= 10^1 \times 0{,}5000 + 10^{-3} \times 0{,}1525 = 10^1 \times 0{,}5000 \\
&\vdots \quad \vdots \quad \vdots \\
S = S_{100} = S_{99} + x_{100} &= 10^1 \times 0{,}5000 + 10^{-3} \times 0{,}1523 = 10^1 \times 0{,}5000
\end{aligned}$$

Analogamente, sumando de derecha a izquierda se obtiene:

$$\begin{aligned}
S'_2 = x_{100} + x_{99} &= 10^{-3} \times 0{,}1523 + 10^{-3} \times 0{,}1523 = 10^{-3} \times 0{,}3046 \\
&\vdots \quad \vdots \quad \vdots
\end{aligned}$$

$$
\begin{aligned}
S'_{94} = S'_{93} + x_7 &= 10^{-1} \times 0{,}1423 + 10^{-3} \times 0{,}1525 = 10^{-1} \times 0{,}1438 \\
S'_{95} = S'_{94} + x_6 &= 10^{-1} \times 0{,}1438 + 10^{-3} \times 0{,}1646 = 10^{-1} \times 0{,}1454 \\
S'_{96} = S'_{95} + x_5 &= 10^{-1} \times 0{,}1454 + 10^{-3} \times 0{,}8229 = 10^{-1} \times 0{,}1536 \\
S'_{97} = S'_{96} + x_4 &= 10^{-1} \times 0{,}1536 + 10^{-1} \times 0{,}3612 = 10^{-1} \times 0{,}5148 \\
S'_{98} = S'_{97} + x_3 &= 10^{-1} \times 0{,}5148 + 10^{1} \times 0{,}1000 = 10^{1} \times 0{,}1051 \\
S'_{99} = S'_{98} + x_2 &= 10^{1} \times 0{,}1051 + 10^{1} \times 0{,}1964 = 10^{-1} \times 0{,}3015 \\
S' = S'_{100} = S'_{99} + x_1 &= 10^{1} \times 0{,}3015 + 10^{1} \times 0{,}1999 = 10^{1} \times 0{,}5014
\end{aligned}
$$

Comparando los resultados $S = 5{,}000$ (suma con i variando de 1 a 100) con $S' = 5{,}014$ (suma con i variando de 100 a 1) resulta evidente que su diferencia relativa (2,8 %) es ampliamente más grande que la diferencia relativa máxima correpondiente a una representación flotante de cuatro dígitos decimales (0,1 %). Vemos entonces que el orden en el que se realiza la suma afecta sensiblemente el resultado final (Imaginar una operación similar, pero con 10^7 términos!).

El resultado "exacto" obtenido luego de operar con gran precisión es: 5,0144815, que redondeado a la precisión utilizada en los cálculos anteriores da 5,014. Esto significa que en este caso, a pesar de la escasa precisión utilizada, pudo igual obtenerse un resultado excelentemente preciso, provista una buena elección del orden de las operaciones.

En la aritmética de precisión finita, todas las propiedades de las operaciones pueden dejar de ser estrictamente válidas en ciertas circunstancias. En el ejemplo anterior se puso en evidencia que los resultados pueden variar al conmutar los sumandos. Adicionalmente, otras propiedades de las operaciones fundamentales pueden también fallar.

Veamos un ejemplo con la propiedad distributiva. Sean $u = 20{,}000$, $v = -6$, $w = 6{,}003$. Se desea calcular $R = uv + uw$ con aritmética decimal de cuatro cifras significativas. Se verifica

$$
\begin{aligned}
(u \times v) &= (10^5 \times 0{,}2000) \times (-10^1 \times 0{,}6000) = -10^6 \times 0{,}1200 \\
(u \times w) &= (10^5 \times 0{,}2000) \times (10^1 \times 0{,}6003) = 10^6 \times 0{,}1201 \\
R &= (-10^6 \times 0{,}1200) + (10^6 \times 0{,}1201) = 10^3 \times 0{,}1000
\end{aligned}
$$

Si, en cambio, se calcula $R' = u(v + w)$, se obtiene

$$
\begin{aligned}
(v + w) &= (-10^1 \times 0{,}6000) + (10^1 \times 0{,}6003) = 10^{-2} \times 0{,}3000 \\
R' &= (10^5 \times 0{,}2000) \times (10^{-2} \times 0{,}3000) = 10^2 \times 0{,}6000
\end{aligned}
$$

El valor exacto es $\overline{R} = 20{,}000 \times (-6 + 6{,}003) = 60$, que en este ejemplo coincide con R', pero difiere substancialmente del resultado R del cálculo original.

Estrategias para corregir estos errores. En general, los errores en resultados finales asignables a la precisión finita de la aritmética pueden ser un problema cuando

se deben realizar cadenas de cálculo que involucran un elevado número de operaciones (millones, o más). En todos los casos es necesario analizar detalladamente el problema particular que se trata, y desarrollar una estrategia para mejorar los resultados imprecisos. Los puntos más importantes a tener en cuenta son:

- Adecuada formulación analítica del problema, con expresiones finales adaptadas para su tratamiento numérico.
- Agrupar convenientemente. Evitar restas de cantidades de similar valor absoluto.
- Redefinir variables cuando sea conveniente.
- Utilizar la máxima precisión disponible siempre que sea posible.

5. Números almacenados en computadoras

Como es sabido, las computadoras electrónicas tienen la capacidad de almacenar información en su memoria. La memoria, a los fines de comprender los conceptos de esta sección, puede entenderse como una sucesión de posiciones las cuales pueden estar en dos estados posibles, "apagadas" o "encendidas". Uno de estos estados lo entenderemos como 0 y el otro como 1. Conceptualmente, la memoria es entonces una sucesión de ceros y unos que pueden ser cambiados según convenga.

Cero y uno son los dígitos disponibles en base dos, y es por lo tanto el sistema de numeración binario el que más naturalmente encuadra con el sistema electrónico de almacenamiento de datos. Los dígitos 0 y 1 reciben el nombre de *dígitos binarios.* Un ejemplo de número expresado en representación binaria es el siguiente: $(110110001010)_2$, equivalente a

$$1 \times 2048 + 1 \times 1024 + 1 \times 256 + 1 \times 128 + 1 \times 8 + 1 \times 2 = (3466)_{10}. \qquad (41)$$

Debido a la facilidad de operaciones de conversión con la base dos, es frecuente el uso de bases $b' = 2^p$ (p entero), en particular $b' = 8$ (numeración octal) y $b' = 16$ (numeración hexadecimal).

En la representación octal, los dígitos utilizados son 0, 1, 2, 3, 4, 5, 6, 7. Así, por ejemplo el número $(9)_{10}$ se representa como $(11)_8$ en el sistema octal. Por su parte, en la representación hexadecimal los dígitos cubren el rango desde 0 hasta $b-1 = 16-1 = 15$; y en este caso los diez dígitos decimales usuales son complementados con las primeras letras del alfabeto, usualmente escritas en mayúsculas, para generar todos los símbolos necesarios, es decir: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Así, $(15)_{10} = (\mathrm{F})_{16}$, $(19)_{10} = (13)_{16}$, $(30)_{10} = (1\mathrm{E})_{16}$, etc.

La conversión entre las mencionadas bases y la base dos es directa, tal como lo ilustra el siguiente ejemplo:

$$(3466)_{10} = \begin{cases} (\quad 6 \quad\quad 6 \quad\quad 1 \quad\quad 2 \quad)_8 \\ (\overbrace{1\ 1\ 0}\ \overbrace{1}\ \overbrace{1\ 0}\ \overbrace{0\ 0}\ \overbrace{1}\ \overbrace{0\ 1\ 0} \quad)_2 \\ (\underbrace{\quad\quad}_{\text{D}} \quad\quad \underbrace{\quad\quad\quad}_{8} \quad\quad \underbrace{\quad\quad}_{\text{A}} \quad)_{16} \end{cases} \tag{42}$$

Puede demostrarse sencillamente (utilizando los algoritmos de conversión descriptos en la sección 2) que $(3466)_{10} = (6612)_8 = (\text{D8A})_{16}$. El cuadro (42) pone en evidencia que los cambios de base de un número binario a base 2^p pueden realizarse agrupando los dígitos de p en p (agregando ceros a la izquierda cuando fuere necesario), y convirtiendo directamente cada uno de los bloques a base 2^p. En los casos $p = 3, 4$ (octal y hexadecimal, respectivamente), las equivalencias son las siguientes:

Dec.	**Bin.**	**Octal**
0	000	0
1	001	1
2	010	2
3	011	3
4	100	4
5	101	5
6	110	6
7	111	7

Dec.	**Bin.**	**Hexadec.**	**Dec.**	**Bin.**	**Hexadec.**
0	0000	0	8	1000	8
1	0001	1	9	1001	9
2	0010	2	10	1010	A
3	0011	3	11	1011	B
4	0100	4	12	1100	C
5	0101	5	13	1101	D
6	0110	6	14	1110	E
7	0111	7	15	1111	F

5.1. Formatos usuales de almacenamiento

5.1.1. Números enteros

Sea z entero. Su representación en base b es,

$$z = s\,(c_{M-1}c_{M-2}\ldots c_1c_0)_b. \tag{43}$$

Usualmente estos números se almacenan en representación binaria con longitud fija. Las longitudes usuales son 16 o 32 dígitos binarios, llamados comunmente *bits*. Estas longitudes corresponden, respectivamente, a 2 o 4 *bytes* (1 byte = 8 bits). En algunos casos se emplean longitudes de 8 bytes (64 bits).

Si la longitud reservada para almacenar el número es de p bits, entonces el número de combinaciones posibles es 2^p. En la casi totalidad de los sistemas, estas combinaciones se utilizan para almacenar los números enteros comprendidos en el intervalo $[-(2^{p-1}-1), 2^{p-1}-1]$, de la siguiente manera.

- Los números positivos se almacenan con su representación binaria habitual:

$$\begin{array}{ccc} 0 & \longrightarrow & \underbrace{0\,0\,0\,\ldots\,0\,0}_{p \text{ bits}} \\ 1 & \longrightarrow & \underbrace{0\,0\,0\,\ldots\,0}_{p-1 \text{ bits}}\,1 \\ \vdots & \vdots & \vdots \\ 2^{p-1}-1 & \longrightarrow & 0\,\underbrace{1\,1\,\ldots\,1\,1}_{p-1 \text{ bits}} \end{array}$$

- Los números negativos se almacenan "a continuación" de los positivos utilizando los *complementos de dos* de sus valores absolutos. El complemento de dos de un número n, $C_2(n)$, se obtiene a partir de la representación binaria $(|n|)_2$ (completada con ceros a la izquierda hasta llenar p dígitos) reemplazando los ceros por unos y viceversa, y luego sumando 1 al resultado,[2] se obtiene:

$$\begin{array}{ccc} -(2^{p-1}-1) & \longrightarrow & 1\,\underbrace{0\,0\,\ldots\,0}_{p-2 \text{ bits}}\,1 \\ \vdots & \vdots & \vdots \\ -2 & \longrightarrow & \underbrace{1\,1\,1\,\ldots\,1}_{p-1 \text{ bits}}\,0 \\ -1 & \longrightarrow & \underbrace{1\,1\,1\,\ldots\,1\,1}_{p \text{ bits}} \end{array}$$

  Este método de representación de los números negativos presenta ventajas técnicas operativas con respecto al sistema "natural" signo + mantisa, y por esta razón es el que se adopta habitualmente.

Vemos entonces que el sistema descripto en el párrafo anterior permite almacenar todos los números enteros z tales que $|z| \leq z_{\max} = 2^{p-1} - 1$. La siguiente tabla enumera las configuraciones más habituales.

p (bits)	Bytes	$z_{\max}$
16	2	32,767
32	4	2,147,483,647
64	8	$9{,}223372031 \times 10^{18}$

5.1.2. Números reales

Sea r real. Su expresión flotante normalizada en base b está dada por la ecuación (5). Usualmente estos números se almacenan en representación binaria o hexadecimal de

[2]Por ejemplo, sea $p = 4$, $n = -6$, $|n| = (0110)_2$. Al revertir los dígitos se obtiene $(1001)_2$, y sumando 1 resulta $C_2(-6) = (1010)_2$.

longitud total p bits. En ambos casos, el signo del número se representa explícitamente destinándose un bit a tal fin. La característica y la mantisa se reparten los $p-1$ bits restantes.

Representación binaria. Cuando $b = 2$ todos los dígitos $d_1, \ldots, d_k$ de la mantisa representada en la ecuación (5) son cero o uno, excepto d_1 que no puede ser cero, y que, por lo tanto, es siempre 1. Por este motivo no resulta necesario almacenar este dígito ya que puede sobreentenderse que es siempre 1. La representación binaria usual de un número en punto flotante es, entonces, de la siguiente forma:

$$\begin{array}{l} 1\,\underbrace{0\,1\,\ldots\,0\,1}\,\underbrace{0\,1\,0\,1\,1\,1\,\ldots\,0\,1} \\ \downarrow \quad \text{exponente} \qquad\quad \text{mantisa} \\ s \quad p_e \text{ bits} \qquad\quad p_m{-}1 \text{ bits} \end{array} \tag{44}$$

donde p_e (p_m) representa el número de bits reservados para la característica (mantisa). Se verifica $p = 1 + p_e + (p_m - 1) = p_e + p_m$.

Para evitar el problema de trabajar con exponentes negativos, se introduce un sezgo en las características almacenadas, es decir, la cantidad que en realidad se guarda es $M + M_0$, siendo M_0 un entero positivo denominado *sezgo* de la representación, al cual se lo elige de modo que la condición $M + M_0 \geq 0$ sea válida en todo el conjunto de números que se desea almacenar.

Si se dispone de p_e bits para el exponente, entonces se pueden almacenar los números enteros pertenecientes al intervalo $[0, 2^{p_e} - 1]$. Tomando, por ejemplo, $M_0 = 2^{p_e-1}$, es posible cubrir el rango de exponentes $-(2^{p_e-1} - 1) \leq M \leq 2^{p_e-1} - 1$, que corresponden a $1 \leq M + M_0 \leq 2^{p_e} - 1$. El caso $M + M_0 = 0$ se reserva para representar al número cero.

Así definido, este sistema de representación de números reales cubre todos los números cuyos valores absolutos están comprendidos en el intervalo $[r_{\min}, r_{\max}]$ (más el número cero), siendo

$$\begin{array}{rcl} r_{\min} & = & 2^{1-M_0}(0{,}100\ldots0)_2 \\ r_{\max} & = & 2^{(2^{p_e}-1)-M_0}(0{,}111\ldots1)_2 \end{array} \tag{45}$$

Las configuraciones más usuales se listan en la siguiente tabla.

Nombre	**Bytes**	p_e (bits)	p_m (bits)	M_0	$1 - M_0$	$(2^{p_e} - 1) - M_0$
Simple prec.	4	8	24	128	-127	127
Doble prec. 1	8	8	56	128	-127	127
Doble prec. 2	8	11	53	1024	-1023	1023
Cuádruple (Cray)	16	11	96	1024	-1023	1023

Para calcular $r_{\min}$ y $r_{\max}$ es necesario tener en cuenta que $2^{127} \simeq 2 \times 10^{38}$ y que $2^{1023} \simeq 9 \times 10^{307}$.

Representación hexadecimal. En representación flotante hexadecimal todos los dígitos de la mantisa del segundo miembro de la ecuación (5) pertenecen al intervalo $[0, 15]$, con excepción de d_1 que debe ser estrictamente mayor que cero.

En algunas computadoras fabricadas en el pasado[3] se almacenan números en punto flotante utilizando su representación en base 16. En esta representación, el número de bits de la mantisa, p_m, debe ser múltiplo de 4 para permitir su división en dígitos hexadecimales. Si P_m es el número de dígitos hexadecimales de la mantisa, entonces $p_m = 4P_m$. En todos los otros aspectos esta representación es similar a la representación binaria. La longitud total necesaria para almacenar un número es $p = 4P_m + p_e + 1$ (bits).

Las configuraciones más usuales del sistema flotante hexadecimal se listan en la siguiente tabla.

Nombre	**Bytes**	p_e (bits)	P_m (d'igitos)	M_0	$1 - M_0$	$(2^{p_e} - 1) - M_0$
Simple prec.	4	7	6	64	-63	63
Doble prec.	8	7	14	64	-63	63

Para calcular el rango $[r_{\min}, r_{\max}]$ cubierto por las representaciones hexadecimales, es necesario tener en cuenta que

$$\begin{aligned} r_{\min} &= 16^{1-M_0}(0{,}100\ldots0)_{16} \\ r_{\max} &= 16^{(2^{p_e}-1)-M_0}(0{,}\mathrm{FFF}\ldots\mathrm{F})_{16}. \end{aligned} \tag{46}$$

Notar que $16^{63} \simeq 7 \times 10^{75}$.

5.2. Precisión de representaciones en punto flotante

ε de una representación. Se define el ε de una representación flotante dada como el mínimo número positivo tal que sumado a la unidad arroje un resultado significativamente diferente en esa representación.

Esta variable nos da una estimación del error relativo con el que se almacenan los números reales.

Consideremos por ejemplo una representación binaria de p_m dígitos de mantisa. Entonces debe ser

$$1 + \varepsilon = 2^1(0{,}1\underbrace{00\ldots0}_{p_m-1})_2 + 2^1(0{,}\underbrace{00\ldots0}_{p_m-1}1)_2 \tag{47}$$

y por lo tanto

$$\varepsilon = 2^{1-p_m}. \tag{48}$$

[3]El caso más emblemático es la en su época famosísima IBM 360.

En forma totalmente análoga, se puede mostrar que en un sistema hexadecimal con mantisa de P_m dígitos, resulta $\varepsilon = 16^{1-P_m}$.

La siguiente tabla contiene los valores de ε teóricos correspondiente a los sistemas de la sección 5.1.2.

Nombre	**Base**	**Dígitos de la mant.**	ε
Simple prec.	2	24	$1{,}2 \times 10^{-7}$
Doble prec. 1	2	56	$2{,}8 \times 10^{-17}$
Doble prec. 2	2	53	$2{,}2 \times 10^{-16}$
Cuádruple	2	96	$2{,}5 \times 10^{-29}$
Simple prec.	16	6	$9{,}5 \times 10^{-7}$
Doble prec.	16	14	$2{,}2 \times 10^{-16}$