Representación numérica, tipos de errores y cifras significativas

Números, errores y cifras

Camilo Cubides
eccubidesg@unal.edu.co

Research Group on Artificial Life – Grupo de investigación en vida artificial – (Alife)
Departamento de Ingeniería de Sistemas e Industrial
Facultad de Ingeniería
Universidad Nacional de Colombia

(Intersemestral 2016)

Agenda

1. Números de máquina
2. Codificación de números enteros y reales
   - Codificación de números enteros
   - Codificación de números reales
3. Tipos de errores
4. Cifras significativas

Contenido

Representación de números reales

Representación de números reales en base b

Todo número real x puede ser representado en un sistema numérico de base $b \in \mathbb{N}$, $b > 1$ en la forma:

$$\begin{aligned} x &= \pm(d_p d_{p-1} \cdots d_2 d_1 d_0 . d_{-1} d_{-2} \cdots d_{-q} \cdots)_b \qquad (1) \\ &= \pm(d_p \times b^p + d_{p-1} \times b^{p-1} + \cdots + d_2 \times b^2 + d_1 \times b^1 + d_0 \times b^0 + \\ &\qquad d_{-1} \times b^{-1} + d_{-2} \times b^{-2} + \cdots + d_{-q} \times b^{-q} + \cdots) \end{aligned}$$

donde $d_i \in \{0, 1, 2, \ldots, b-1\}$, para cada $i = p, p-1, p-2, \ldots, 2, 1, 0, -1, -2, \ldots, -q, \ldots$.

Example

$$+745.86_{10} = +(7 \times 10^2 + 4 \times 10^1 + 5 \times 10^0 + 8 \times 10^{-1} + 6 \times 10^{-2})$$

$$-101.011_2 = -(1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3})$$

Representación de punto flotante

Dado un número real x representado como en la expresión (1), el número x se puede escribir equivalentemente en la forma:

$$\pm 0.a_1a_2a_3\cdots \times b^e, \tag{2}$$

donde $a_i \in \{0, 1, 2, \ldots, b-1\}$ para cada $i \in \mathbb{N}^+$, b es la base y $e \in \mathbb{Z}$. Esta representación se llama de "representación de punto flotante de x para la base b", e se llama *exponente* o *característica* y la cadena $M = a_1a_2a_3\cdots$ se llama *mantisa*. Por lo anterior, todo número real se puede representar de forma compacta así $\pm 0.M \times b^e$ con $0 \leq 0.M \leq 1$.

Remark

Si la base es 2 al número se le dice que está escrito en binario, en el caso de 10 se dice que está escrito decimal, en caso de 8 se dice que está escrito en octal, en caso de 16 se dice que está escrito en hexadecimal. Estas son las principales bases en las que se representan números reales.

Forma normal

Representación de punto flotante normalizada

Dado que un número en punto flotante, este puede expresarse de distintas formas que son equivalentes, es necesario establecer una única representación. Es por ello que se trabaja con *números normalizados*.

Si $x \neq 0$, entonces, mediante operaciones en el exponente, es posible encontrar una única representación de x tal que la primera cifra de la mantisa no sea cero, es decir, que el dígito a la izquierda del punto está entre 0 y la base b $(0 < a_1 < b)$.

Ejemplo de la forma normal de número reales

Definition

Sea x un número real, si $x = \pm a_1.a_2a_3a_4 \cdots \times b^e$ y $0 < a_1 < b$ entonces a la expresión $\pm a_1.a_2a_3a_4 \cdots \times b^e$ se le conoce como la *forma normal* de x.

En particular, se dice que un número binario está normalizado si el dígito a la izquierda del punto es igual a 1.

Example

- La forma normal de 891.246_{10} es $8.91246_{10} \times 10^2$.
- La forma normal de 0.00753_{10} es $7.53_{10} \times 10^{-3}$.
- La forma normal de 1011.001_2 es $1.011001_2 \times 2^3 = 1.011001_2 \times 2^{11_2}$.

Conversión de decimal a binario

Theorem

Para convertir un número real en base 10 a base 2, se debe convertir su parte entera a base 2, luego su parte fraccionaria a base 2, sumarlas y por último colocar el signo del número original.

Example

Para convertir el número real decimal -118.8125_{10} a binario, primero se debe hallar la representación binaria del número 118_{10}, luego la del número 0.8125_{10}, se suman estos valores y por último se le coloca el signo $-$. A continuación se explica cómo se pueden hacer esas conversiones.

Conversión de decimal entero a binario entero

Theorem (Algoritmo de la división o división euclídea)

Sean $m, n \in \mathbb{Z}$, con $n > 0$. Existen dos enteros únicos q y r tales que:

$$m = q \times n + r \tag{3}$$

donde $0 \leq r < n$. Al entero q se le llama el cociente y a r el residuo.

Regla para convertir un decimal entero a binario entero

Dado un entero m_{10}, su expresión en base 2 se obtiene dividiendo m_{10} por 2 tantas veces como sea posible hasta obtener cero (0) como cociente. Esto es equivalente a expresar m_{10} en la forma $m_{10} = q \times 2 + r$ y todos los cocientes de la misma forma, hasta obtener el cociente cero (0). La nueva representación del número se obtiene escribiendo los residuos en secuencia de forma inversa a como se obtuvieron.

Ejemplo de conversión de entero a binario

Example (continuation)

$$
\begin{aligned}
118 &= 59 \times 2 + \mathbf{0} \\
59 &= 29 \times 2 + \mathbf{1} \\
29 &= 14 \times 2 + \mathbf{1} \\
14 &= 7 \times 2 + \mathbf{0} \\
7 &= 3 \times 2 + \mathbf{1} \\
3 &= 1 \times 2 + \mathbf{1} \\
1 &= 0 \times 2 + \mathbf{1}
\end{aligned}
\qquad \uparrow
$$

Result

$$118_{10} = 1110110_2$$

Regla para convertir un decimal fraccionario a binario fraccionario

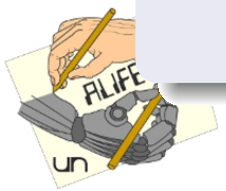
Dado un real x_{10}, $0 < x_{10} < 1$, su expresión en base 2 se obtiene multiplicando x_{10} por 2 o la parte fraccionaria de cada producto encontrado tantas veces como sea posible hasta obtener uno (1.0) como producto o el número de cifras suficientes para alcanzar la precisión deseada. La nueva representación del número fraccionario se obtiene escribiendo en secuencia las partes enteras de cada producto que se halla obtenido.

Example (continuation)

$$
\begin{aligned}
0.8125 \times 2 &= \mathbf{1}.625 \\
0.625 \times 2 &= \mathbf{1}.25 \\
0.25 \times 2 &= \mathbf{0}.5 \\
0.5 \times 2 &= \mathbf{1}.0
\end{aligned}
\qquad \downarrow
$$

Result

$$0.8125_{10} = 0.1101_2$$



Example (continuation)

Como $118_{10} = 1110110_2$ y $0.8125_{10} = 0.1101_2$, entonces:

$$\begin{aligned} -118.8125_{10} &= -\left(1110110_2 + 0.1101_2\right) \\ &= -1110110.1101_2 \end{aligned}$$

Exercise

Expresar el número decimal 93.25_{10} en binario.

Conversión de binario a decimal

Theorem

Para convertir un número real x_2 en base 2 a base 10, se expresa el número como sumas de potencias de base 2 tales como en la expresión (1), se realizan las operaciones indicadas, y el valor obtenido junto con el signo del número original es la representación del número x_2 en decimal.

Example

Para convertir el número real binario -1110110.1101_2 a decimal, se expresa este número así:

$$\begin{aligned} 1110110.1101_2 = & \\ & (1 \times 2^6) + (1 \times 2^5) + (1 \times 2^4) + (0 \times 2^3) \\ & + (1 \times 2^2) + (1 \times 2^1) + (0 \times 2^0) \\ & + (1 \times 2^{-1}) + (1 \times 2^{-2}) + (0 \times 2^{-3}) + (1 \times 2^{-4}) \end{aligned}$$

Example (continuation)

de lo anterior se tiene que:

$$\begin{aligned} 1110110.1101_2 &= 2^6 + 2^5 + 2^4 + 2^2 + 2^1 + 2^{-1} + 2^{-2} + 2^{-4} \\ &= 64 + 32 + 16 + 4 + 2 + \frac{1}{2} + \frac{1}{4} + \frac{1}{16} \\ &= 118 + \frac{13}{16} = 118 + 0.8125 = \\ &= 118.8125_{10} \end{aligned}$$

de donde

$$-1110110.1101_2 = -118.8125_{10}$$

Exercise

Expresar el número binario $11110001.\overline{01}_2$ en decimal. Ayuda: recuerde que la suma de la serie geométrica $\sum_{n=0}^{\infty} cr^n$ converge a: $\frac{c}{1-r}$, si $|r| < 1$.

Números de máquina

Definition (Números de máquina)

El conjunto $\mathbb{M}$ de los números en forma de punto flotante normalizada, que pueden ser representados en un computador, se llama *conjunto de números de máquina*. $\mathbb{M}$ depende de la base b, de la longitud de la mantisa $m > 0$, y el rango para el exponente $e \in \{-k, -k+1, \ldots, K-1, K\}$, siendo $k, K \in \mathbb{N}^+$. Explícitamente,

$$\begin{aligned}\mathbb{M} &= \mathbb{M}(b, m, k, K) \\ &= \{0\} \cup \Big\{ \pm a_1.a_2a_3\cdots a_m \times b^e : a_1, a_2, \ldots, a_m \in \{0, 1, \ldots, b-1\}, \\ &\qquad a_1 \neq 0, e \in \{-k, -k+1, \ldots, K-1, K\}\Big\}\end{aligned}$$

El conjunto $\mathbb{M}$ es un conjunto finito ($|\mathbb{M}| < \omega$), que está contenido como subconjunto propio del conjunto de lo números racionales ($\mathbb{M} \subset \mathbb{Q}$).

Overflow y *underflow*

Con base en lo anteriormente tratado, se tiene que el conjunto de números de máquina es finito, por lo que en la representación de los números reales se encuentran huecos entre los distintos números.

Definition

Cuando se intenta almacenar un número más grande que el número máximo representable se produce un evento especial llamado *overflow*. Del mismo modo, cuando se intenta almacenar un número más pequeño que el mínimo representable se produce un evento especial llamado *underflow*.

Para estas dos zonas el tratamiento es especial, dependiendo del programa compilador a lenguaje de máquina, el lenguaje usado para programar. Principalmente, un overflow y un underflow se pueden tratar mediante el redondeo hacia números que sí sean representables, o simplemente se genera una excepción o un error.

Example

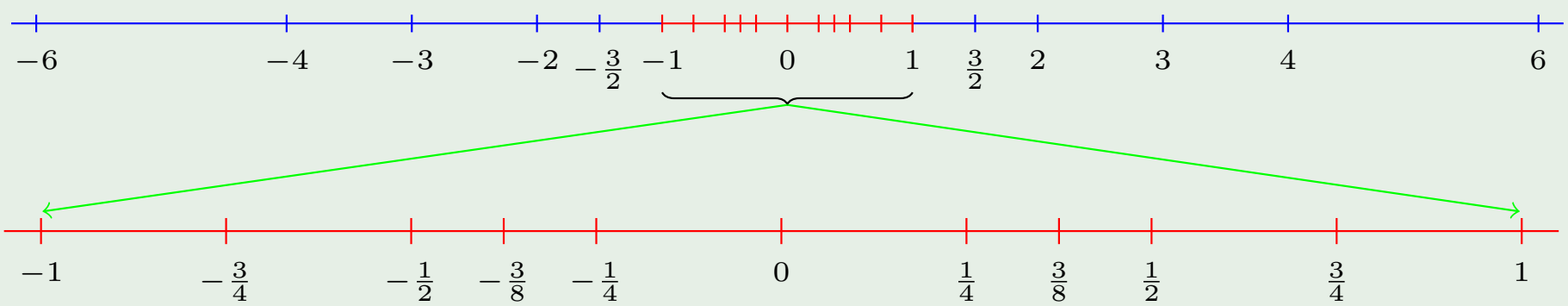
Construir explícitamente la máquina correspondiente a $\mathbb{M}(2,2,2,2)$. Calcular el número positivo más pequeño y el más grande.

Solution

Teniendo en cuenta la definición de números de máquina, se tiene que:

$$\mathbb{M} = \{0\} \cup \left\{ \pm 1.d \times 2^e \mid d \in \{0,1\}, e \in \{-2,-1,0,1,2\} \right\} =$$

$$\left\{ -6, -4, -3, -2, -\frac{3}{2}, -1, -\frac{3}{4}, -\frac{1}{2}, -\frac{3}{8}, -\frac{1}{4}, 0, \frac{1}{4}, \frac{3}{8}, \frac{1}{2}, \frac{3}{4}, 1, \frac{3}{2}, 2, 3, 4, 6 \right\}$$



Solution (continuation)

$$x^+_{min} = 1.0_2 \times 2^{-2} = \left[\left(1 \times 2^0\right) + \left(0 \times 2^{-1}\right)\right] \times 2^{-2} = \frac{1}{4}$$

$$x^+_{max} = 1.1_2 \times 2^{2} = \left[\left(1 \times 2^0\right) + \left(1 \times 2^{-1}\right)\right] \times 2^{2} = 6$$

- underflow se presenta en el conjunto $(-\frac{1}{4}, 0) \cup (0, \frac{1}{4})$.
- overflow se presenta en el conjunto $(-\infty, -6) \cup (6, \infty)$.

Epsilon de máquina

Definition (Epsilon de máquina)

El epsilon de máquina es el número decimal más pequeño que sumado a 1 se puede representar de manera precisa en la máquina (que no es redondeado), es decir, retorna un valor diferente de 1, éste da una idea de la precisión o número de cifras almacenadas

Proposition

En el sistema binario el épsilon de máquina es igual a 2^{-n}, donde n es la longitud de la mantisa sin tomar en cuenta el bit implícito antes del punto flotante.

Contenido

Contenido

Signo y magnitud

Todo número entero se puede representar en la forma $n = (-1)^s \times m$, donde $s \in \{0, 1\}$ y $m \in \mathbb{N}$. En la codificación del tipo *signo y magnitud*, se utiliza un arreglo de k bits, el primer bit "el más significativo (1)" sirve para guardar el valor de s y el resto de bits $\big(2, \ldots, n\big)$ se utilizan para guardar la representación binaria de número natural m, escribiéndolo hacia el extremo derecho del arreglo.

Se adopta la convención de que todos los números cuyo bit del extremo izquierdo sea cero (0) son positivos, y por ende, todos aquellos cuyo bit del extremo izquierdo sea uno (1) son negativos, así como se muestra a continuación:

Signo y magnitud (continuation)

Examples

Para $k = 8$ la representación de los números $+23$ y -23 es:

$$+23_{10} = +10111_2 = \underbrace{\boxed{0}}_{\text{bit del signo}}\underbrace{\boxed{0}\boxed{0}\boxed{1}\boxed{0}\boxed{1}\boxed{1}\boxed{1}}_{\text{bits de la magnitud}}$$

$$-23_{10} = -10111_2 = \underbrace{\boxed{1}}_{\text{bit del signo}}\underbrace{\boxed{0}\boxed{0}\boxed{1}\boxed{0}\boxed{1}\boxed{1}\boxed{1}}_{\text{bits de la magnitud}}$$

Signo y magnitud (continuation)

Existe una desventaja a la hora de utilizar la representación signo y magnitud, ésta es que existen dos representaciones diferentes del número cero (0).

Examples

Para $k = 8$ la representación de los números $+0$ y -0 es:

$$+0_{10} = +0_2 = \underbrace{\boxed{0}}_{\text{bit del signo}}\underbrace{\boxed{0}\boxed{0}\boxed{0}\boxed{0}\boxed{0}\boxed{0}\boxed{0}}_{\text{bits de la magnitud}}$$

$$-0_{10} = -0_2 = \underbrace{\boxed{1}}_{\text{bit del signo}}\underbrace{\boxed{0}\boxed{0}\boxed{0}\boxed{0}\boxed{0}\boxed{0}\boxed{0}}_{\text{bits de la magnitud}}$$

Complemento a 1

Otra manera de representar números negativos es conocida como *complemento a 1*, ésta se define como:

Definition

El *complemento a* $b-1$ de un número $n \geq 0$, representado en k dígitos en base b se define como:

$$C_{b-1}(n_b) = \left(b-1_1 \cdots b-1_k\right) - n_b$$

donde $b-1$ es el valor máximo de un dígito en base b.

Examples

$$C_9(13579_{10}) = 99999_{10} - 13579_{10} = 86420_{10}$$
$$C_1(01101_2) = 11111_2 - 01101_2 = 10010_2$$

Complemento a 1 (continuation)

Remark

El complemento a 1 de un número n escrito en código binario n_2 se obtiene invirtiendo cada bit de n_2.

Example

$$C_1(10010110111_2) = 01101001000_2$$

Complemento a 1 (continuation)

Definition

Utilizando la representación en complemento a 1 se tiene que los números positivos se representan en forma usual (como en signo y magnitud) y los negativos como el complemento a 1 del valor absoluto del número.

Por lo tanto, se adopta la convención de que todos los números cuyo bit del extremo izquierdo sea 0, son positivos; y por ende, todos aquellos cuyo bit del extremo izquierdo se 1 son negativos.

Complemento a 1 (continuation)

Example

Si $C_1(86) = C_1(001010110_2) = 110101001_2$, entonces la representación del número positivo 86_{10} es 001010110_2 y la del número negativo -86_{10} es el complemento a 1 de 86_{10}, es decir, 110101001_2.

Existe una desventaja a la hora de utilizar la representación de complemento a 1, es que existen dos posibles representaciones para el número cero (0).

Examples

$$+\mathbf{0_{10}} = 0 \cdots 0_2$$
$$-\mathbf{0_{10}} = 1 \cdots 1_2$$

Complemento a 2

Otra manera de representar números negativos que no conlleva el problema de la doble representación del 0 es conocida como *complemento a 2*, ésta se define en general como:

Definition

El *complemento a b* de un número $n \geq 0$, representado en k dígitos en base b se define como:

$$C_b(n_b) = (10_1 \cdots 0_k) - n_b$$

Examples

$$C_10(13579_{10}) = 100000_{10} - 13579_{10} = 86421_{10}$$
$$C_2(01101_2) = 100000_2 - 01101_2 = 10011_2$$

Complemento a 2 (continuation)

Se puede demostrar que $C_b(n_b) = C_{b-1}(n_b) + 1$. Así

Example

$$C_2(01101_2) = C_1(01101_2) + 00001_2 = 10010_2 + 00001_2 = 10011_2$$

Remark

El complemento a 2 de un número n escrito en código binario n_2 se obtiene copiando de derecha a izquierda todos los bits de n_2 hasta encontrar el primer 1 inclusive, e invertir todos los bits restantes hacia la izquierda.

Example

$$C_2(00110011100_2) = 11001100100_2$$

Complemento a 2 (continuation)

Definition

Utilizando la representación de complemento a 2 se tiene que los números positivos se representan de forma usual (como en signo y magnitud) y los negativos como el complemento a 2 del valor absoluto del número.

Por lo tanto, se adopta la convención de que todos los números cuyo bit del extremo izquierdo sea 0, son positivos; y por ende, todos aquellos cuyo bit del extremo izquierdo sea 1 son negativos.

Complemento a 2 (continuation)

Examples

- Si $C_2(86_{10}) = 110101010_2$ entonces la representación del número negativo -86_{10} es el complemento a 2 de 86_{10}, es decir, 110101010_2.
- $C_2(+00000000_{10}) = 100000000_2 - 00000000_2 = 100000000_2$, si se utilizan sólo 8 bits para almacenar el número entonces $C_2(+00000000_{10}) = 00000000_2$, así que la representación del número -00000000_{10} es 00000000_2, la cual es la misma que signo magnitud para 00000000_{10}, de donde

$$+\mathbf{0_{10}} = -\mathbf{0_{10}} = 0 \cdots 0_2$$

Exceso

Otro mecanismo para representar números negativos es conocido como exceso a $2^{k-1} - 1$.

Definition

Si se utilizan k bits para representar enteros, entonces, dado un entero n, la representación por exceso de n es:

$$ex(n) = n + 2^{k-1} - 1$$

Example

Si para representar un número entero se dispone de 8 bits, entonces la representación del número -19 está dado por

$$-19 + 2^{8-1} - 1 = -19 + 128 - 1 = 108_{10} = 01101100_2$$

Exceso (continuation)

Remark

Con este tipo de representación, los números enteros representables pertenecen al conjunto $\{-(2^{k-1}-1),\ldots,0,\ldots,2^{k-1}\}$. Si para representar un número entero se dispone de 8 bits, entonces la representación del cero (0_{10}) seria 01111111_2, el -127_{10} seria 00000000_2 y para el 128 seria 11111111_2.

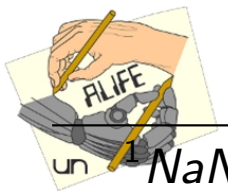
Como se observa el número 2^{k-1} no posee inverso aditivo y el cero (0_{10}) tiene una representación única en el rango de representatividad del sistema de codificación por exceso.

Contenido

Estándar IEEE 754

En 1985, el Instituto papa Ingenieros Eléctricos y Electrónicos, IEEE (*Institute for Electrical and Electronic Engineers*) publico un informe llamado *IEEE Standard for Binary Floating-Point Arithmetic (IEEE 754-1985)*. Se especificaron los formatos para la precisiones simple, doble y extendida para la representación de números reales de punto flotante, incluyendo el cero y valores especiales (infinito y NaN)[1].

Dentro del estándar IEEE 754 se especifica que el formato para la representación de valores en coma flotante de precisión simple se utilizan 32 bits, para la de doble precisión se utilizan 64 bits y para la extendida se utilizan usualmente 80 bits.



[1] *NaN: Not a Number*

Representación normalizada de la mantisa extendida

Definition

Dado un número real x representado en *forma normal,* $x := \pm 0.a_1a_2a_3\cdots \times b^e$, con $a_1 \neq 0$. Entonces, la representación se puede modificar de tal manera que halla un único dígito a la izquierda del punto de la parte fraccionaria distinto de cero, a este formato se le llama *representación normalizada de la mantisa extendida*.

Dada la representación normalizada de x, ésta se puede modificar de tal manera que x adquiera el formato normalizado de la mantisa extendida, simplemente corriendo hacia la derecha el punto de la parte fraccionaria, así: $x = \pm a_1.a_2a_3\cdots \times b^{e-1}$.

Representación normalizada de la mantisa extendida para números binarios I

Si el número real x se representa en forma binaria, entonces se tiene que el formato normalizado de la mantisa extendida de x adquiere la forma: $x = \pm 1.a_2a_3 \cdots a_i \cdots \times 2^e$, con $a_i \in \{0, 1\}$ para cada $i \geq 2$ y $e \in \mathbb{Z}$

Representación normalizada de la mantisa extendida para números binarios II

Por la observación anterior se tiene que x se puede escribir genéricamente en forma binaria de la siguiente manera

$$x = \pm 1.a_2a_3\cdots a_i\cdots \times 2^e = (-1)^s \times (1+f) \times 2^{c-2^{k-1}+1}$$

donde $s \in \{0,1\}$, $f = 0.a_2a_3\cdots a_i\cdots \in (0,1)$, con $a_i \in \{0,1\}$ para cada $i \geq 2$ y $c = e + 2^{k-1} - 1$.

Como el primer dígito de la mantisa extendida, se sabe que implícitamente es 1, entonces no es necesario almacenarlo. Por esta razón, sólo los valores s, f y c (en binario) se utilizan para codificar el número en punto flotante.

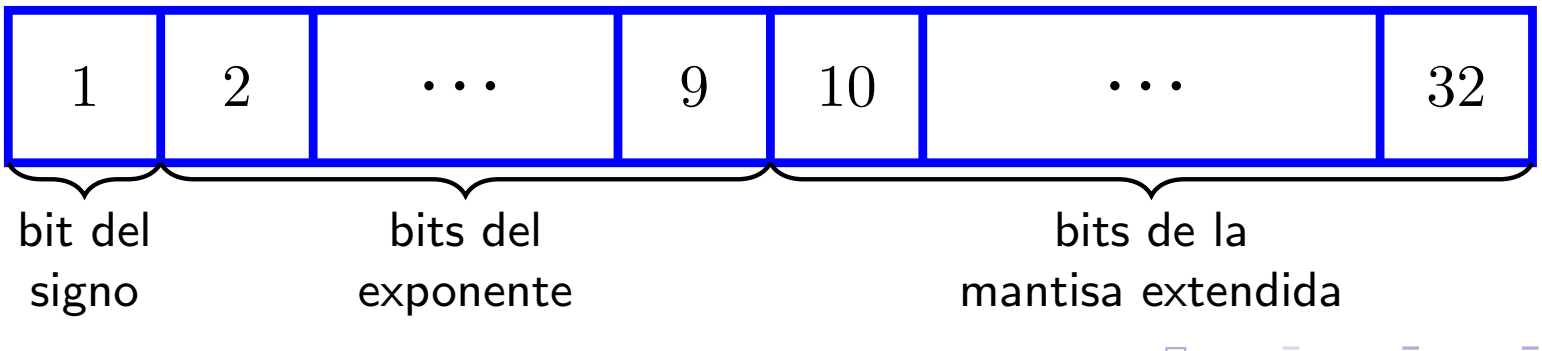
Representación de valores en formato de precisión simple

Para representar un número x en formato de precisión simple se utilizan 32 bits de la siguiente manera:

- El número x se escribe en la forma

$$x = (-1)^s \times (1+f) \times 2^{c-2^{8-1}+1} = (-1)^s \times (1+f) \times 2^{c-127}.$$

- El primer bit se usa para el signo (s). $\langle 1 \rangle$.
- Los siguientes 8 bits se utilizan para el exponente (c) representado en exceso a $2^{8-1} - 1 = 127$. $\langle 2 - 9 \rangle$.
- Los restantes 23 bits para la mantisa extendida (f). $\langle 10 - 32 \rangle$. !Realmente la mantisa se representa con 24 bits, por que el primer bit, que siempre es uno, se encuentra implícito¡.



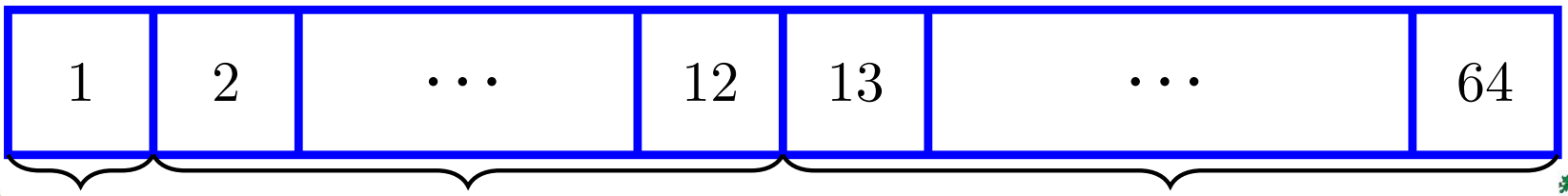
Representación de valores en formato de doble precisión

Para representar un número x en formato de precisión simple se utilizan 64 bits de la siguiente manera:

- El número x se escribe en la forma

$$x = (-1)^s \times (1+f) \times 2^{c-2^{11-1}+1} = (-1)^s \times (1+f) \times 2^{c-1023}.$$

- El primer bit se usa para el signo (s). $\langle 1 \rangle$.
- Los siguientes 11 bits se utilizan para el exponente (c) representado en exceso a $2^{11-1} - 1 = 1023$. $\langle 2 - 12 \rangle$.
- Los restantes 52 bits para la mantisa extendida (f). $\langle 13 - 64 \rangle$. !Realmente la mantisa se representa con 53 bits, por que el primer bit, que siempre es uno, se encuentra implícito¡.



Example

Un año gaussiano es una unidad de tiempo definida como 365.2568983 días. Fue adoptada por Carl Friedrich Gauss como la duración del año sidéreo durante sus estudios de la dinámica del Sistema Solar.

Para la representación en formato de precisión simple, primero se convierte el valor de un año gaussiano a número binario de la siguiente manera:

1. $365_{10} = 101101101_2$
2. $0.2568983_{10} = 0.010000011100010000001011\cdots_2$
3. $365.2568983_{10} \approx 101101101.01000001110001 0_2$,
   sólo se toman las primeras 24 cifras a partir del primer 1 a la izquierda, pues esa es la máxima longitud de la mantisa que se puede almacenar en un número de precisión simple.

Example (continuation)

4. Ahora, escribiendo el número $101101101.01000001110001 0_2$ en la forma $(-1)^s \times (1+f) \times 2^{c-127}$ se tiene que:

$$
\begin{aligned}
101101101.01000001110001 0_2 = & \\
& (-1)^0 \times 1.01101101010000011100010_2 \times 2^8 = \\
& (-1)^0 \times 1_2 + 0.01101101010000011100010_2 \times 2^{135-127} = \\
& (-1)^0 \times 1_2 + 0.01101101010000011100010_2 \times 2^{10000111_2-127}
\end{aligned}
$$

Por lo anterior se tiene que la representación en formato de precisión simple de un año gaussiano es:

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

Exercise

Suponga que se "extiende" el estándar IEEE 754 a un formato de precisión mini-simple, en la cual un número se codifica con 16 bits, donde el primer bit es para el signo del número, los siguientes 5 bits son para el exponente y los restantes 10 bits son para la mantisa. Codifique mediante el formato de precisión mini-simple, las siguientes aproximaciones a las constantes matemáticas:

- $e \approx 2.7182818$
- $\pi \approx \frac{355}{113} = 3.1415927$

Ahora, decodifique cada uno de los números y observe cual es el valor real almacenado.

Contenido

Clases de errores

Los errores asociados a todo cálculo numérico tiene su origen en dos grandes factores:

1. Aquellos que son inherentes a la formulación del problema.
   - Aquellos en los que la definición matemática del problema es sólo una aproximación a la situación física (p. ej. cuando se descartan variables dentro de un modelo).
   - Aquellos en los que se utilizan constantes físicas y datos empíricos (p. ej. aproximación finita de constantes con expresiones infinitas y errores en los instrumentos de medición).

2. Los que son consecuencias del método empleado para encontrar la solución del problema.

   de bulto: cuando hay errores en cálculos manuales o *bugs* (defecto de software) en los programas utilizados para encontrar las soluciones a los problemas.

   por truncamiento: cuando la solución encontrada a un problema no se ha hecho como se formuló originalmente, sino mediante una aproximación (aproximación de un infinito o infinitesimal mediante un valor finito).

   - Cálculo de una función mediante los primeros n términos de su expansión en serie de Taylor (p. ej. $\sin(x)$).
   - Aproximación de una integral por una suma finita (p. ej. el método de Simpson).
   - Resolución de una ecuación diferencial reemplazando las derivadas por una aproximación (p. ej. las diferencias finitas).
   - Solución de una ecuación $f(x) = 0$ por un método iterativo (p. ej. el método de Newton-Raphson).

de redondeo: cuando los cálculos numéricos no pueden realizarse mediante precisión ilimitada, porque muchos números requieren infinitos decimales para ser representados correctamente (p. ej. π) o en algunas operaciones se sobrepasa la cantidad de decimales que son representables (p. ej. como en la multiplicación). Por esta razón, para operar con ellos es necesario redondearlos, es decir, se descartan los dígitos a partir de una posición de la parte fraccionaria.

Error absoluto real y error relativo real

Definition (Error absoluto real)

Suponga que p^* es una aproximación de p. El *error absoluto real* $E_a(p)$ de la aproximación p se define como:

$$E_a(p) = |p - p^*|$$

Definition (Error relativo real)

Suponga que p^* es una aproximación de p. El *error relativo real* $E_r(p)$ de la aproximación p se define como:

$$E_r(p) = \left|\frac{E_a(p)}{p}\right| = \left|\frac{p - p^*}{p}\right|$$

si $p \neq 0$.

Example

Suponga que se tiene que medir la longitud de un puente y la de un remache, y se obtiene 9999 cm y 9 cm, respectivamente. Si los valores verdaderos son 10000 cm y 10 cm, calcular

1. el error absoluto real
2. el error relativo real

Solution

1. El error absoluto real en la medición del puente es:

$$|10000 \text{ cm} - 9999 \text{ cm}| = 1 \text{ cm}$$

El error absoluto real en la medición del remache es:

$$|10 \text{ cm} - 9 \text{ cm}| = 1 \text{ cm}$$

Solution

2. El error relativo real en la medición del puente es:

$$\left|\frac{10000\ \text{cm} - 9999\ \text{cm}}{10000\ \text{cm}}\right| = \left|\frac{1\ \text{cm}}{10000\ \text{cm}}\right| = 0.0001\ \text{cm}$$

El error relativo real en la medición del remache es:

$$\left|\frac{10\ \text{cm} - 9\ \text{cm}}{10\ \text{cm}}\right| = \left|\frac{1\ \text{cm}}{10\ \text{cm}}\right| = 0.1\ \text{cm}$$

Aunque ambas medidas tiene un error absoluto real igual $(1\ \text{cm})$, el error relativo real del remache $(0.1\ \text{cm} \equiv 10\%)$ es mucho mayor que la del puente $(0.0001\ \text{cm} \equiv 0.01\%)$. En conclusion, la medida hecha del puente es más exacta que la del remache.

Error absoluto y error relativo con métodos iterativos

Suponga que durante un cálculo se ha obtenido la sucesión $x_0, x_1, \ldots, x_n$ de aproximaciones a la solución de un problema, que se desconoce en todo momento, entonces el error absoluto iterativo y el error relativo iterativo para cada $k > 0$ se definen como:

Definition (Error absoluto iterativo)

$$E_a = E_a(x_k, x_{k-1}) = |x_k - x_{k-1}|$$

Definition (Error relativo iterativo)

$$E_r = E_r(x_k, x_{k-1}) = \left|\frac{E_a}{x_k}\right| = \left|\frac{x_k - x_{k-1}}{x_k}\right|$$

si $x_k \neq 0$.

Redondeo por eliminación y redondeo por aproximación

Sea $x = d_n d_{n-1} \cdots d_1 d_0 . d'_1 d'_2 \cdots d'_k d'_{k+1} \cdots$, un número real escrito en base b y coma flotante.

Definition (Redondeo por eliminación)

El número obtenido al realizar *redondeo por eliminación* en la k-ésima cifra de la parte fraccionaria del numero x es:

$$rd_e(x) = d_n d_{n-1} \cdots d_1 d_0 . d'_1 d'_2 \cdots d'_k$$

Cota del error relativo real del redondeo por eliminación

Una cota del error relativo real que se comete por el uso de redondeo por eliminación $rd_e(x)$, de un número en base b, coma flotante y k cifras significativas esta dado por:

$$E_r\big(rd_e(x)\big) \leq b^{1-k}$$

Problem

Sea $x = 0.98568723$ un número real en base 10, coma flotante, realizar el redondeo por eliminación a 5 cifras de la parte fraccionaria.

Solution

1. $rd_e(0.98568723) = 0.98568$

Example

Supóngase que se realiza un cálculo en una máquina y se obtiene un número x representado en base 10, coma flotante, y se tienen cinco (5) cifras para representar la parte fraccionaria. Si se realiza un redondeo por aproximación, entonces una cota para el error relativo real debido al redondeo vale:

$$E_r\big(rd_e(x)\big) \leq 10^{1-5} = 10^{-4} = 0.0001$$

Definition (Redondeo por aproximación)

El número obtenido al realizar *redondeo por aproximación* en la k-ésima cifra de la parte fraccionaria del numero x es:

$$rd_a(x) = \begin{cases} d_n d_{n-1} \cdots d_1 d_0 . d'_1 d'_2 \cdots d'_k, & \text{si } d'_{k+1} < \frac{b}{2}; \\ d_n d_{n-1} \cdots d_1 d_0 . d'_1 d'_2 \cdots d'_k + b^{(-k)}, & \text{si } d'_{k+1} \geq \frac{b}{2}. \end{cases}$$

Cota del error relativo real del redondeo por aproximación

Una cota del error relativo real que se comete por el uso de redondeo por aproximación $rd_a(x)$, de un número en base b, coma flotante y k cifras significativas esta dado por:

$$E_r\left(rd_a(x)\right) \leq \frac{1}{2} \times b^{1-k}$$

Problem

Sea $x = 0.98568723$ un número real en base 10, coma flotante, realizar el redondeo por aproximación a 5 cifras de la parte fraccionaria.

Solution

1. $rd_a(0.98568723) = 0.98569$

Example

Supóngase que se realiza un cálculo en una máquina y se obtiene un número x representado en base 10, coma flotante, y se tienen cinco (5) cifras para representar la parte fraccionaria. Si se realiza un redondeo por aproximación, entonces una cota para el error relativo real debido al redondeo vale:

$$E_r\big(rd_a(x)\big) \leq \frac{1}{2} \times 10^{1-5} = \frac{1}{2} \times 10^4 = 0.5 \times 0.0001 = 0.00005$$

Example

Usando el estándar de la IEEE para aritmética en coma flotante (IEEE 754), las cotas para la precisión simple y doble del error relativo real al redondeo por aproximación están dadas por:

- Para precisión simple se tiene que el bit más significativo es para el signo, los 8 siguientes son para el exponente y los 23 restantes para la mantisa. Así que, el error relativo real está dado por:

$$E_r\big(rd_a(x)\big) \leq \frac{1}{2} \times 2^{1-24} = \frac{1}{2} \times 2^{-23} = 2^{-24} \approx 5.96 \times 10^{-8}$$

  el 24 es por que el primer uno (1) de la mantisa no se representa.

- Para precisión doble se tiene que el bit más significativo es para el signo, los 11 siguientes son para el exponente y los 52 restantes para la mantisa. Así que, el error relativo real está dado por:

$$E_r\big(rd_a(x)\big) \leq \frac{1}{2} \times 2^{1-53} = \frac{1}{2} \times 2^{-52} = 2^{-53} \approx 1.11 \times 10^{-16}$$

Contenido

Cifras significativas correctas

Definition

La cantidad de *cifras significativas* de un número son la primera no nula y todas las siguientes. Así 2.350 tiene cuatro cifras significativas, mientras que 0.00023 tiene sólo dos.

El error relativo real está íntimamente relacionado con la noción de cifras significativas correctas. Suponga que x^* es una aproximación de x.

Definition (primera)

Una aproximación x^* a x tiene q cifras significativas correctas si al redondear x^* y x a q cifras, se obtiene el mismo resultado.

Example

Sea $x = 0.9949$ un número real y $x^* = 0.9951$ una aproximación a x, entonces:

- x^* tiene una cifra significativa, ya que $rd_a(x) = 1.0$ y $rd_a(x^*) = 1.0$
- x^* no tiene dos cifras significativas, ya que $rd_a(x) = 0.99$ y $rd_a(x^*) = 1.0$
- x^* tiene tres cifras significativa, ya que $rd_a(x) = 0.995$ y $rd_a(x^*) = 0.995$

Remark

Por la ambigüedad anterior, la primera definición de cifras significativas correctas no es apropiada, por lo tanto es necesario especificar una definición más precisa.

Cifras significativas correctas

Definition (segunda (apropiada))

Una aproximación x^* a x tiene q cifras significativas correctas si el error relativo real verifica que:

$$q = \max\left\{p \in \mathbb{Z}^+ : E_r(x) \leq \frac{1}{2} \times 10^{-p}\right\}$$

Example

Con la definición anterior y sin peligro de ambigüedad, se tiene que $x^* = 0.9951$ tiene tres cifras significativas correctas de $x = 0.9949$, ya que:

$$E_r(0.9949) = \left|\frac{0.9949 - 0.9951}{0.9949}\right| = \left|-\frac{0.0002}{0.9949}\right| = \left|\frac{0.0002}{0.9949}\right|$$

$$\approx 0.000201 \leq \frac{1}{2} \times 10^{-3} = 0.0005$$

Exactitud y precisión

Definition

La *exactitud* se refiere a qué tan cercano está el valor calculado o medido del valor verdadero.

Definition

La *precisión* se refiere a que tan cercanos se encuentran, unos de otros, diversos valores calculados o medidos.

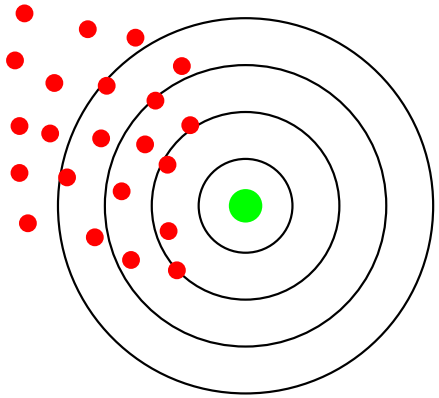
Definition

La *inexactitud* o *sesgo* se define como la desviación sistemática del valor verdadero.

Definition

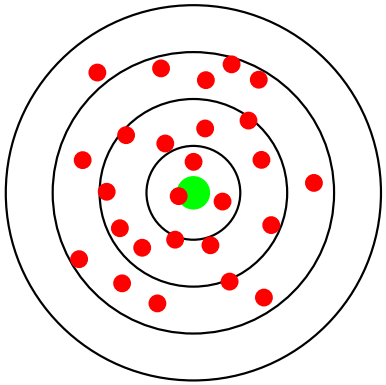
La *imprecisión* o *incertidumbre* se refiere a la magnitud en la dispersión de los valores calculados o medidos.

Comparación entre exactitud y precisión de valores



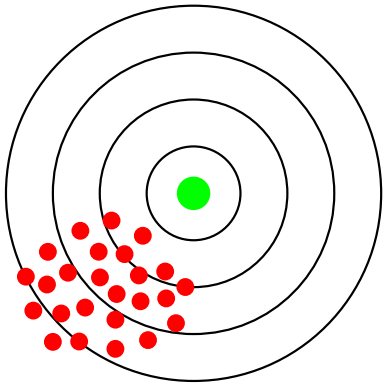
inexacto
impreciso

a)



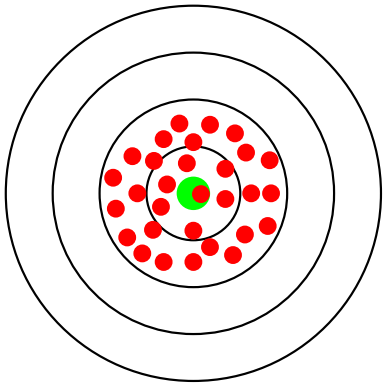
exacto
impreciso

b)



inexacto
preciso

c)



exacto
preciso

d)