Número, algoritmo y errores

Índice

Veamos que pasa si ejecutamos la siguientes instrucciones con Matlab

```
>> a = 7/pi
>> a*pi-7
```

¿Qué resultado deberíamos obtener?

¿Qué resultado obtenemos?

¿Y si calculamos sin(pi)?

¿Obtenemos el resultado esperado?

Los resultados anteriores son debidos a la existencia de **errores numéricos de almacenamiento**.

Debido al hecho inevitable que los ordenadores digitales sólo pueden almacenar una cantidad finita de información, se producen errores numéricos por el mero hecho de almacenar un número en general.

Aunque algunos números pueden ser almacenados exactamente, en general al almacenar un número en un ordenador cometeremos **errores de almacenamiento o redondeo**.

En el ejemplo que hemos visto:

```
>> a = 7/pi
>> a*pi-7
```

Matlab almacena el número pi con un número finito de decimales cometiendo un **error de almacenamiento**.

Naturalmente, al efectuar operaciones con este número, el resultado se ve afectado por estos errores también y por eso no obtenemos cero como se esperaba.

Además, los resultados se vuelven a almacenar en el ordenador por lo que de nuevo se cometen errores de almacenamiento. Por lo tanto, cuando se opera con un ordenador, se comenten errores de almacenamiento que a su vez se van propagando (**propagación de errores**).

En general, los resultados proporcionados por los ordenadores son muy precisos, pero **siempre debemos tener presente la existencia de los errores de almacenamiento y su propagación** y asegurarnos que estos errores son despreciables en relación a la precisión con la que necesita obtener los resultados.

En la página

http://www.ima.umn.edu/~arnold/disasters/disasters.html

pueden encontrarse algunos ejemplos de problemas causados por la propagación del error en la resolución de problemas en ingeniería.

Fallo del Misil Patriot (Guerra del Golfo, 1991)

Durante la guerra del golfo, un misil americano falló al intentar interceptar un misil iraquí.

Motivo: se calculó con poca precisión el tiempo interno del misil.

El ataque del misil iraquí mató a 28 soldados e hirió unas 100 personas.

El reloj interno medía en décimas de segundo, y para calcular el tiempo en segundos se multiplicaba por 1/10.

El número 1/10 es un número periódico en binario, y al almacenar éste número se cometió un error de redondeo de 0.000000095_{10}.

La bateria del misil Patriot llevaba unas 100 horas funcionando que medido en décimas de segundo son 3600000.

Al pasar este tiempo a segundos, se cometió un error de

$$3600000 \cdot 0.000000095 = 0.342 \text{ segundos}$$

En este tiempo, el misil iraquí recorrió más de medio quilómetro respecto de la posición prevista de intercepción.

En este tema entenderemos el origen de dichos errores.

Aunque no será objetivo de este curso, es importante además, tratar de **cuantificarlos** con el objetivo de controlarlos, esto es, obtener **cotas del error**.

Errores

Definiciones

- Error absoluto y relativo

  x valor exacto y $\overline{x}$ valor aproximado

  - Error absoluto

$$E_x = x - \overline{x}$$

  - Error relativo

$$r_x = \frac{x - \overline{x}}{x}$$

Ejemplo: considerad la aproximación 3.14 del número pi. Comprovad que el error absoluto es 0.00159 y el error relativo es 0.000507.

- Cifras significativas
  - Las cifras significativas de un número son la primera cifra no nula i las siguientes
  - La aproximación x a $\bar{x}$ tiene q cifras significativas correctas si el error relativo verifica

$$|r_x| \leq \frac{1}{2}10^{-q}$$

Ejemplo: 3.14 tiene 3 cifras significativas.
¿Cuántas cifras significativas correctas tiene la aproximación 3.14 del número pi?
El error relativo es $0.000507 \leq 0.005 \cdot 10^{-2}$

Números

Representación de un número en base n

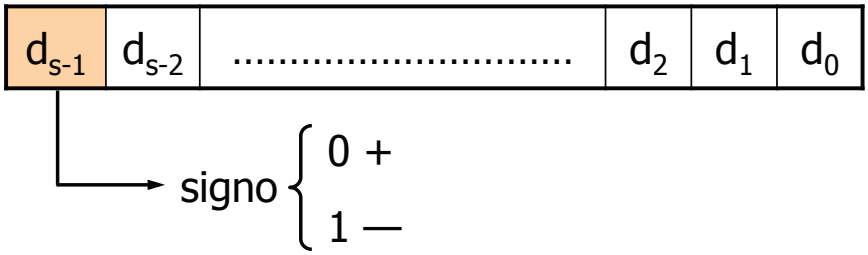
$$(d_p\, d_{p-1} \ldots d_1 d_0 . d_{-1} \ldots d_{-(q-1)}\, d_{-q})_n =$$
$$= d_p n^p + d_{p-1} n^{p-1} + \ldots + d_1 n + d_0 n^0 + d_{-1} n^{-1} + \ldots + d_{-(q-1)}\, n^{-(q-1)} + d_{-q} n^{-q}$$

$(124.5)_{10} = 1 \cdot 10^2 + 2 \cdot 10^1 + 4 \cdot 10^0 + 5 \cdot 10^{-1}$
$(101.11)_2 = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2}$

En un ordenador los números se guardan en binario

Almacenamiento de números enteros

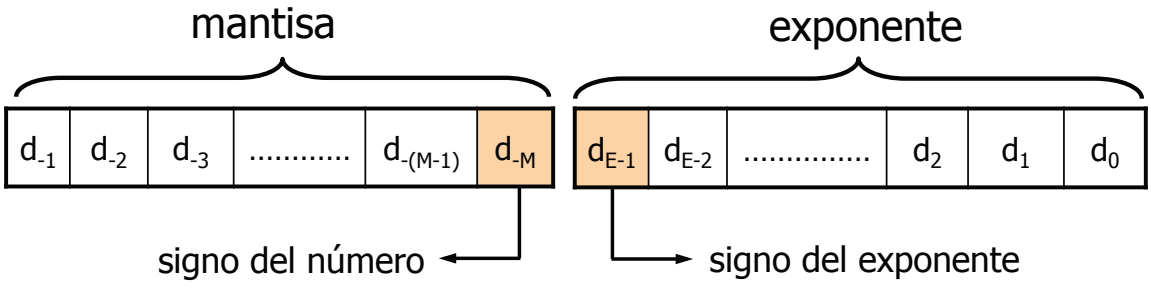
Los números enteros se almacenan utilizando s posiciones de memoria (bits) de las cuales se reserva una para el signo.



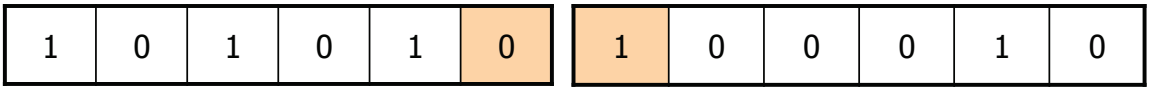
$$|N_{max}| = 2^{s-2} + \ldots + 2^2 + 2 + 1 = 2^{s-1} - 1$$

Almacenamiento de números reales

- base 2
- coma flotante
- M posiciones para la mantisa
- E posiciones para el exponente



Ejemplo: $(0.10101)_2 \cdot 2^{-(10)_2} = (2^{-1}+2^{-3}+2^{-5}) \cdot 2^{-2} = 0.1640625$



$$|N_{max}| = \left(2^{-1} + 2^{-2} + \ldots + 2^{-(M-1)}\right) 2^{\left(2^{E-2} + \ldots + 2^1 + 2^0\right)} =$$

$$= \left(1-2^{(1-M)}\right) 2^{\left(2^{E-1} -1\right)}$$

1	1	1	.....	1	1 0	0	1	.....	1	1	1

$$|N_{min}| = \left(2^{-1}\right) 2^{-\left(2^{E-2} + \ldots + 2^1 + 2^0\right)} =$$

$$= 2^{-\left(2^{E-1}\right)}$$

1	0	0	.....	0	1 0	1	1	.....	1	1	1

Almacenamiento en Matlab

En Matlab se usa el formato **double** que usa 64 bits totales:
- 11 para el exponente (E=11)
- 53 para la mantisa (M=53)

Por lo tanto, los números máximos y mínimos que se pueden almacenar con Matlab son:

$$|N_{max}| = (1-2^{(1-53)}) \cdot 2^{(2^{11-1} -1)} < 2^{1023} \approx 0.899 \cdot 10^{308}$$

$$|N_{min}| = 2^{-(2^{11-1})} = 2^{-1024} \approx 0.556 \cdot 10^{-308}$$

Underflow y overflow

Overflow → error que se produce cuando se intenta almacenar un número más grande que el máximo permitido por el tipo de número usado

Underflow → error que se produce cuando se intenta almacenar un número menor que el mínimo permitido

Ejecutad el programa potencias2.m para comprobar el overflow y underflow de Matlab.

Obtenemos los resultados esperados?

El programa potencias2.m pide por pantalla un entero n y escribe en el fichero potencias2.res las potencias

2^i i=0...n si n es positivo

2^i i =0:-1:n si n es negativo

¿Qué valores máximos y mínimos de n deberíamos poder calcular y qué obtenemos?

$|N_{max}| < 2^{1023}$ $|N_{min}| = 2^{-1024}$

Mejoras de Matlab:

- Overflow: no se guarda el 1 de la coma flotante 0.**1**01
- Underflow: permite poner ceros en la mantisa $2^{-52} \cdot 2^{-1022}$

0	0	0	.....	0	0	1	1 0	-1022

Error relativo de almacenamiento

En Matlab los números de almacenan en base 2 con 53 bits para la mantisa (sin contar el signo) y 11 bits para el exponente.

¿Cuál es el error relativo que se comete al almacenar un número mediante truncamiento?

Número: $(. d_{-1}\, d_{-2}\, d_{-3} \ldots d_{-52}\, d_{-53}\, d_{-54}\, d_{-55})_2 \cdot 2^{exp}$

d_{-1}	d_{-2}	d_{-3}	.....	d_{-51}	d_{-52}	d_{-53}	1 0	exp.

Número: $(. d_{-1}\, d_{-2}\, d_{-3} \ldots d_{-52}\, d_{-53}\, d_{-54}\, d_{-55} \ldots)_2 \cdot 2^{exp}$

d_{-1}	d_{-2}	d_{-3}	.....	d_{-51}	d_{-52}	d_{-53}	1 0	exp.

Error relativo:

$$|r_x| = \frac{(. 0\,0\,0 \ldots 0\,0\, d_{-54}\, d_{-55} \ldots)_2 \cdot 2^{exp}}{(. d_{-1}\, d_{-2}\, d_{-3} \ldots d_{-52}\, d_{-53}\, d_{-54}\, d_{-55} \ldots)_2 \cdot 2^{exp}}$$

$$= \frac{(. d_{-54}\, d_{-55} \ldots)_2 \cdot 2^{exp} \cdot 2^{-53}}{(. d_{-1}\, d_{-2}\, d_{-3} \ldots d_{-52}\, d_{-53}\, d_{-54}\, d_{-55} \ldots)_2 \cdot 2^{exp}}$$

$$= \frac{(. d_{-54}\, d_{-55} \ldots)_2}{(. d_{-1}\, d_{-2}\, d_{-3} \ldots d_{-52}\, d_{-53}\, d_{-54}\, d_{-55} \ldots)_2} \cdot 2^{-53}$$

$$|r_x| = \frac{(.\, d_{-54}\, d_{-55}\, \ldots\,)_2}{(.\, d_{-1}\, d_{-2}\, d_{-3}\, \ldots\, d_{-52}\, d_{-53}\, d_{-54}\, d_{-55} \ldots\,)_2} \cdot 2^{-53}$$

$$< \frac{(.\, 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ \ldots\,)_2}{(.100000\ldots001111111)_2} \cdot 2^{-53}$$

$$= \frac{2^{-1} + 2^{-2} + 2^{-3} + 2^{-4} + \ldots}{2^{-1} + 2^{-54} + 2^{-55} + 2^{-56} + 2^{-57} + \ldots} \cdot 2^{-53}$$

$$= \frac{1 + 2^{-1} + 2^{-2} + 2^{-3} + \ldots}{1 + 2^{-53} \cdot (1 + 2^{-1} + 2^{-2} + \ldots)} \cdot 2^{-53}$$

$$= \frac{2}{1 + 2^{-53} \cdot 2} \cdot 2^{-53} = \frac{1}{1 + 2^{-54} \cdot 2} \cdot 2^{-52} =$$

$$= 2.22 \cdot 10^{-16}$$

Error relativo de almacenamiento

De la misma forma, se obtiene que el error relativo máximo que se comete cuando se almacena un número utilizando la técnica de aproximación es:

$$|r_x| \leq 1.11 \cdot 10^{-16}$$

Mirar el fichero suma_1_eps.m para comprobar que en Matlab:

$$1 + 2^{-53} = 1 + 1.11 \cdot 10^{-16} = 1$$

ZERO DE MÁQUINA: $1.11 \cdot 10^{-16}$

Propagación de errores

Hemos visto que al almacenar un número en Matlab se comete un **error de almacenamiento** del orden de 10^{-16}.

Naturalmente, al efectuar operaciones con este número, el resultado se ve afectado por estos errores también.

Además, los resultados se vuelven a almacenar en el ordenador por lo que de nuevo se cometen errores de almacenamiento. Por lo tanto, cuando se opera con un ordenador, se comenten errores de almacenamiento que a su vez se van propagando (**propagación de errores**).

Por lo tanto, cuando implementemos un programa en Matlab, nos tendremos que asegurar que estos errores que se van propagando, son despreciables en relación a la precisión con la que necesita obtener los resultados.

Algoritmo bien / mal condicionado:

Un algoritmo puede abstraerse como un operador que toma unos datos a y devuelve un resultado b.

Si afectamos los datos de un error, δa, el resultado será $b+\delta b$.

Un algoritmo está bien condicionado si $|\delta b|<M|\delta a|$ donde $M>0$ es una constante no muy grande.

Algoritmo

Es importante tener en cuenta que dos algoritmos matemáticamente equivalentes no tienen porque serlo en aritmética finita.

Comprobad que dan los siguientes comandos:

>> 1e16 * ((1 – 1) + 1e-16) ans = 1

>> 1e16 * ((1+ 1e-16) – 1) ans = 0

Por lo tanto es importante vigilar que **los algoritmos usados no amplifiquen los errores**.

Clasificación de los errores

- Error inherente
  Propio de los datos con con los que se inician los cálculos.
- Error de redondeo
  Al trabajar con aritmética finita, sólo es posible almacenar un número finito de dígitos
- Error de truncamiento
  Un algoritmo ha de tener un número finito de operaciones

Algoritmo

Se quieren calcular las sucesivas potencias del inverso del número áureo utilizando diferentes algoritmos.

$$\Phi = \frac{\sqrt{5}-1}{2} = 0{,}61803\ldots$$

El primero consiste en multiplicar la potencia anterior por Φ

1

$$\begin{aligned} \Phi^0 &= 1 \\ \Phi^k &= \Phi \cdot \Phi^{k-1} \qquad k \geq 1 \end{aligned}$$

Teniendo en cuenta que Φ es raíz de la ecuación

$$x^2 = 1 - x$$

sus potencias se pueden calcular como

2

$$\begin{aligned} \Phi^0 &= 1 \\ \Phi^1 &= \Phi \\ \Phi^k &= \Phi^{k-2}(1-\Phi) \qquad k \geq 2 \end{aligned}$$

Y aplicando la propiedad distributiva al producto anterior se puede escribir un tercer algoritmo para calcular las potencias de Φ

3

$$\begin{aligned} \Phi^0 &= 1 \\ \Phi^1 &= \Phi \\ \Phi^k &= \Phi^{k-2} - \Phi^{k-1} \qquad k \geq 2 \end{aligned}$$

Ejecutad los programas aureo1.m, aureo2.m y aureo3.m y mirad las diferencias entre los resultados obtenidos.

Propagación de errores

Material adicional

x, y valores exactos de dos números
$\overline{x}$, $\overline{y}$ valores aproximados

$$E_x = x - \overline{x} \qquad E_y = y - \overline{y}$$

$$r_x = \frac{E_x}{x} = \frac{x - \overline{x}}{x} \qquad r_y = \frac{E_y}{y} = \frac{y - \overline{y}}{y}$$

- Propagación del error en la suma

$$s = x + y \qquad \overline{s} = \overline{x} + \overline{y}$$

$$E_s = s - \overline{s} = E_x + E_y$$

$$r_s = \frac{E_s}{s} = \frac{x}{x+y} r_x + \frac{y}{x+y} r_y$$

- Propagación del error en la resta

$$r = x - y \qquad \overline{r} = \overline{x} - \overline{y}$$

$$E_r = r - \overline{r} = E_x - E_y$$

$$r_r = \frac{E_r}{r} = \frac{x}{x-y} r_x - \frac{y}{x-y} r_y$$

- Propagación del error en el producto

$$p = xy \qquad \overline{p} = \overline{xy}$$

$$E_p = p - \overline{p} \approx xE_y + yE_x$$

$$r_p = \frac{E_p}{p} \approx r_x + r_y$$

Propagación del error en la división

$$d = x/y \qquad \overline{d} = \overline{x}/\overline{y}$$

$$E_d = d - \overline{d} \approx \frac{yE_x - xE_y}{y^2}$$

$$r_d = \frac{E_d}{d} \approx r_x - r_y$$

Propagación del error en una función

$$z = f(x) \qquad \overline{z} = f(\overline{x})$$

$$E_z = z - \overline{z} \approx f'(x)E_x$$

$$r_z = \frac{E_z}{z} \approx x\frac{f'(x)}{f(x)}r_x$$