ANÁLISIS DE ERRORES Y SU PROPAGACIÓN

1. INTRODUCCIÓN A LA TEORÍA DE ERRORES Y SU PROPAGACIÓN

Cuando se trata de determinar el valor de una magnitud, el número que se obtiene como resultado de las medidas no es el valor exacto de dicha magnitud, sino que estará afectado por un cierto error debido a múltiples factores. Hablando en términos generales, se llama error de una medida a la diferencia entre el valor obtenido y el valor real de la magnitud medida. Si, repitiendo la experiencia, medimos varias veces la misma magnitud, obtendremos cada vez un valor distinto y se nos plantea el problema de decidir cuál de todos los valores hallados es el que ofrece mayores garantías de exactitud. A la resolución de este problema se encamina el contenido de este Capítulo. El que inicia su contacto con la experimentación, debe dejar de lado la idea de que puede obtener el valor exacto de una magnitud física. La premisa fundamental de la que debe partir es que la exactitud total es inalcanzable. Con este punto de arranque y con la ayuda de la teoría de errores, las conclusiones deberían ir surgiendo solas a lo largo de la realización de las prácticas, siendo algunas de ellas:

- El resultado de una medida es de poco valor si no se conoce su precisión.
- La precisión de una medida puede ser en sí misma objeto de estudio.
- El diseño de un experimento incluye el estudio previo de los errores que se cometerán.

Números en un computador

Sea x un número real positivo. La representación decimal normalizada de x en un computador, con k cifras significativas es

$$x = 0.d_1d_2...d_k \times 10^n$$

donde d_i es un entero en el intervalo [0, 9] y d1 ≥ 1. El valor k, los valores mínimo y máximo permitidos para n dependen del computador, del sistema operativo o del lenguaje. Una manera aproximada de obtener estos valores en Matlab es la siguiente:

format(30)

x = 1/3 El resultado es

0.333333333333333148296

Únicamente hay 16 dígitos correctos, los demás son "basura" producida por Matlab para satisfacer el formato deseado. Esto nos indica que en Matlab, en la representación interna de un número, no hay más de 16 cifras significativas.

Relacionado con el concepto anterior, está el épsilon de la máquina, que se define así:

$$\varepsilon_{maq} = \min\{t > 0 : 1 + t \neq 1\}$$

La anterior definición usa los números utilizados en el computador. Este conjunto de números es finito y la definición tiene sentido. Obviamente si los valores *t* se tomaran en *R*, el valor épsilon de la maquina estaría mal definido.

Una manera aproximada de obtener el ´épsilon de la maquina consiste en buscar, por ensayo y error, un valor x tal que 1 + x > 1 y 1 + x/10 = 1.

La orden

x = 1.0e-10; x1 = 1+x; x2 = 1+x/10; (x1 > 1) & (x2 == 1)

produce F ("false"), en cambio

x = 1.0e-15; x1 = 1+x; x2 = 1+x/10; (x1 > 1) & (x2 == 1)

produce T ("true"). Esto nos indica que un valor aproximado es justamente 10−15. Al utilizar Matlab tiene un valor predefinido

%eps = 2.220E-16

Para averiguar si un número positivo y pequeño es considerado como nulo, se puede ensayar con diferentes valores de la potencia de 10, por ejemplo:

x = 1.0e-20; x == 0.0

produce como resultado F, indicando que x no es nulo. Al ensayar

x = 1.0e-100; x == 0.0

el resultado de nuevo es F. Después de varios ensayos

x = 1.0e-323; x == 0.0

produce F y

x = 1.0e-324; x == 0.0

produce T, es decir, 10−324 es considerado como nulo.

Para evitar el ensayo y error se puede utilizar la siguiente secuencia de ordenes´

```
x = 1;
while x/10 > 0.0 x0 = x;
   x = x/10; end
x_final = x0
```

El resultado obtenido es 9.881-323. Obsérvese que x toma los valores 1, 1/10, 1/100, ... Sin embargo el resultado obtenido no es exactamente una potencia de 10.

Ahora queremos averiguar qué tan grandes pueden ser los números en Matlab. Así la orden

```
x = 1.0e308
```

muestra en la pantalla 1.000+308, resultado esperado. La orden

```
x = 1.0e309
```

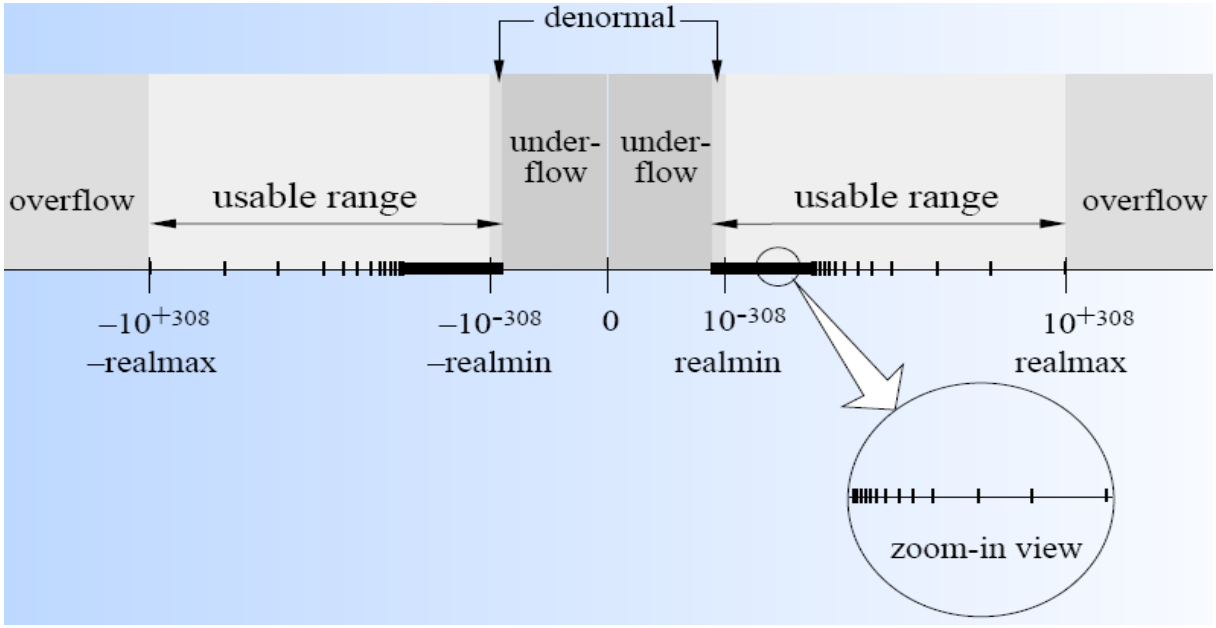
1.1. OVERFLOW Y UNDERFLOW

Los números en formato de doble precisión que aparecen en los cálculos y tienen una magnitud menor que $2.2250738585072014x10^{-308} = 2^{-1023} * (1 + 2^{-52})$, producen un *desbordamiento de la capacidad mínima* o "Underflow" o subdesbordamiento y, por lo general, se igualan a cero.

Los números en formato de doble precisión mayores que

$1.7976931348623157x10^{308} = 2^{1024} * (2 – 2^{-52})$

producen un **"Overflow"** o **desbordamiento** y hacen que se detengan los cálculos (***Run Time Error***). La lógica del uso de dígitos binarios tiende a encubrir las dificultades de cálculo que aparecen al usar una colección finita de números de máquina para representar a todos los números reales.



Matlab trabaja en doble precisión con 64 bits.

Si el número supera el valor “máximo real” en Matlab (1.79769313486231570D+308) sucede un “Overflow” pero no se interrumpe la ejecución y el número almacenará el valor de la variable “%Inf”. (Usar ieee(2)).

pr = number_properties(“huge")

Si el número es menor al valor “mínimo real” en Matlab (2.22507385850720138D-308) sucede un “Underflow” y cualquier valor menor a este se considera cero.

pr = number_properties("tiny")

Para examinar estos problemas, supondremos que los números de máquina se representan en la forma de punto flotante decimal normalizada

$$\pm 0.d_1d_2d_3 \ \ldots \ d_k \times 10^n, \quad 1 \le d_1 \le 9, \quad \text{y} \quad 0 \le d_i \le 9, \quad \text{para cada } i = 2, 3\ldots, k.$$

Los números escritos de esta forma se llaman *números de máquina decimales con k* dígitos.

Cualquier número real positivo, *y*, dentro del intervalo numérico de la máquina se puede normalizar como sigue:

$$y = 0.d_1d_2d_3 \ \ldots \ d_k \, d_{k+1} \, d_{k+2} \ldots \times 10^n$$

1.2. TRUNCAMIENTO Y REDONDEO

Sea x un real (supuesto positivo por facilidad de presentación,

$$x = 0.d_1d_2...d_k \times 10^n$$

su representación normalizada y t un entero positivo menor que k. El número obtenido por truncamiento con t cifras significativas es

$$x' = 0.d_1d_2...d_t \times 10^n$$

Dicho de otra forma, se quitan los últimos $k - t$ dígitos. El redondeo con t cifras significativas se puede presentar de varias maneras equivalentes. Una de ellas es la siguiente,

$$\text{redondeo}(x, t) = \text{truncamiento}\left(x + 0.\underbrace{00\cdots0}_{t-1}5 \times 10^n \right), t)$$

truncamiento(1234.56789, 2) = 1200

truncamiento(1234.56789, 6) = 1234.56

redondeo(1234.56789, 2) = 1200

redondeo(1234.56789, 6) = 1234.57

Una manera sencilla, que funciona cuando $d_t \leq 8$, es la siguiente: los primeros $t - 1$ dígitos son los mismos y el dígito en la posición t es

$$\delta_t = \begin{cases} d_t + 1 \text{ si } d_{t+1} \leq 4 \\ d_t + 1 \text{ si } d_{t+1} \geq 5 \end{cases}$$

Si $d_t = 9$ y $d_{t+1} \leq 4$, entonces $\delta_t = d_t$. Ahora bien, el caso especial se tiene si $d_t = 9$ y $d_{t+1} \geq 5$, entonces se suma 1 a $d_t = 9$, volviéndose 10 y se escribe $\delta_t = 0$, pero hay que agregar ("llevar") 1 al dígito d_{t-1}, etc.

1.3. ERROR RELATIVO Y ABSOLUTO

Si x es un número real y $\tilde{x}$ es una aproximación se definen el error absoluto (siempre no negativo) y el error relativo cuando $x \neq 0$, de la siguiente forma:

error absoluto = $|x - \tilde{x}|$

$$\textbf{error relativo} = \frac{|x - \tilde{x}|}{|x|}$$

Ejemplo. Sean ***x*** y ***y*** números reales, $\tilde{x}$ el redondeo de x con $n = 5$ cifras significativas, $\tilde{y}$ el redondeo de y con n cifras significativas, $z = x - y$, $\tilde{z}$ el redondeo de $\tilde{x} - \tilde{y}$ con n cifras significativas, e_a el error absoluto entre z y $\tilde{z}$, e_r el error relativo.

x	y	$\tilde{x}$	$\tilde{y}$	z	$\tilde{z}$	e_a	e_r
1/7	2/3	0.14286	0.66667	−11/21	−0.52381	4.8e-7	9.1e-7
1/7	0.14284	0.14286	0.14284	0.00001714...	0.00002	2.9e-6	1.7e-1

En el segundo caso, el error relativo es grande, aproximadamente 17%.

Los principales casos en los que los errores pueden ser grandes, son:

1. Suma de cantidades de tamaños muy diferentes.
2. esta de cantidades muy parecidas.
3. División por un número cercano a cero.

Estos casos, en cuanto sea posible, deben ser evitados y, si no es posible, los resultados deben ser interpretados de manera muy cuidadosa.

1.4. ERRORES EN LA ARITMÉTICA DEL COMPUTADOR

Además de la representación imprecisa de los números, la aritmética realizada en una computadora no es exacta. La aritmética implica el manejo de los dígitos binarios mediante diversas operaciones matemáticas o lógicas.

Como la mecánica real de estas operaciones no es pertinente a esta presentación, se utilizará un enfoque propio de la aritmética de una computadora. Aunque la siguiente aritmética no dará la imagen exacta, bastará para explicar los problemas potenciales.

2. ENLACES SUGERIDOS

3. BIBLIOGRAFÍA

Análisis Numérico, *Richard L. Burden/J. Douglas Faires*, Editorial Thomson Learning Inc.

Métodos Numéricos Con SCILAB, Héctor Manuel Mora Escobar, Abril 2010

4. GLOSARIO

5. PREGUNTAS DE AUTOEVALUACIÓN

1. ¿Qué es error relativo?

2. ¿Qué es error absoluto?