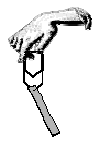
Fuentes de error

Representación de números en la computadora

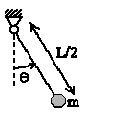
18/03/2015

Modelo



PROBLEMA REAL

-Observación de un fenómeno
-Detección del problema (idealización)
- Referencia a conocimientos previos (físicos y matemáticos)

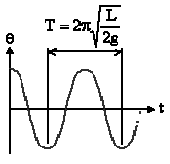


$\frac{d^2\theta}{dt^2}+\frac{2g}{L}\operatorname{sen}\theta=0$

$\theta<<1 \Rightarrow \frac{d^2\theta}{dt^2}+\frac{2g}{L}\theta=0$

MODELO MATEMÁTICO

Permite analizar situaciones reales y preveer comportamientos



RESOLUCION

Procedimientos analíticos o numéricos

CÁLCULO NUMÉRICO

Aplicación de algoritmos computacionales

Confrontación con situaciones experimentales para validar el modelo matemático

Fuentes de error en un modelo

- Datos (ruido, instrumentales)
- Variables consideradas
- Definición del modelo
- Representación de los números reales en la computadora
  - Naturaleza discreta (limitados)
  - Error de redondeo
  - Acumulación de errores
  - Error de desbordamiento

  } Precisión de cálculo
- Algoritmos de métodos Numéricos
  - Errores de aproximación
  - Algoritmos de cálculo inestables

  } Confiabilidad de los resultados
- Problemas mal condicionados

Representación de los números

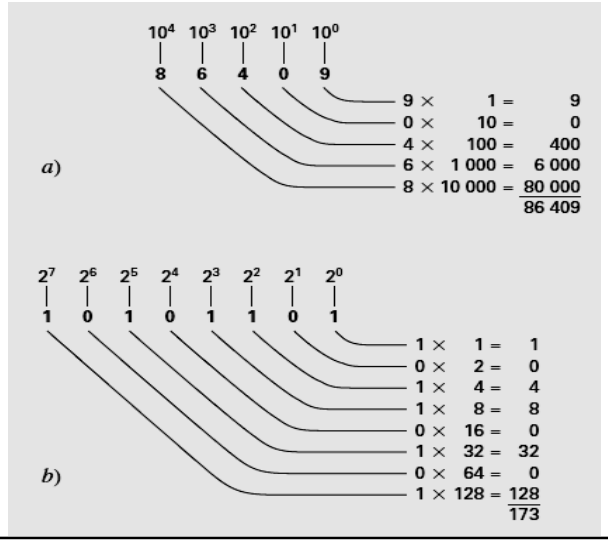
Interacción de computadoras con el usuario ⇒ Cambio de base
Usuario: Base 10 ⇒ Computadora: Base 2

Las computadoras codifican en binario sólo, dos estados, 0 ó 1.

•Representación decimal ó en base 10, dígitos entre 0 y 9.
Número decimal = dígito decimal * potencia de 10, según su posición, ej.:
$528.37_{10} = 5*10^2 + 2*10^1 + 8*10^0 + 3*10^{-1} + 7*10^{-2}$

•Representación binaria ó en base 2, dígitos entre 0 y 1.
Número decimal = dígito binario (bit, **bi**nary digi**t**) * potencia de 2, según su posición, ej.:
$1101.01_2 = 1*2^3 + 1*2^2 + 0*2^1 + 1*2^0 + 0*2^{-1} + 1*2^{-2} = 13.25_{10}$

•Representación hexadecimal ó en base 16, dígitos entre 0 y F (0...9,A,B,C,D,E,F)
Número decimal = dígito hexadecimal (1 byte = 8 bits = 2 dígitos hexa) * potencia de 16, según su posición, ej.:
$6AF.2E_{16} = 6*16^2 + 10*16^1 + 15*16^0 + 2*16^{-1} + 14*16^{-2} = 1711.17968750_{10}$



Aritmética finita

Los números decimales en general se obtienen a partir de otra base como:

$$\sum_{i=-m}^{n} d_i \beta^i$$

β : base
$\mathbf{d_i}$: dígito entre 0 y β -1
n : cantidad de dígitos enteros -1
m: cantidad de decimales

En la computadora la cantidad de dígitos binarios para definir un número es limitada.

Las computadoras usan sólo un **subconjunto finito**, relativamente pequeño, de los números reales para representar a "todos" los números reales.

Representación de números enteros binarios sin signo

Ej.: Números binarios de 4 bits

Binario	Decimal	Binario	Decimal	Binario	Decimal
0000	0	0101	5	1010	10
0001	1	0110	6	1011	11
0010	2	0111	7	1100	12
0011	3	1000	8	1101	13
0100	4	1001	9	1110	14
				1111	15

- Con *n* bits se pueden representar los números enteros en el rango [0, 2^n -1]. (De 0_{10} a 15_{10}, de 0_{16} a F_{16})
- El cantidad de bits limita la cantidad de números que podemos representar.
- Una operación con dos números de *n* bits puede exceder la cantidad de bits necesaria para representarlo.

Representación de números enteros binarios con signo

Ej.: Números binarios de 4 bits
Un bit es destinado al signo.

Binario	Decimal	Binario	Decimal	Binario	Decimal
0000	0	0101	5	1010	-2
0001	1	0110	6	1011	-3
0010	2	0111	7	1100	-4
0011	3	1000	-0	1101	-5
0100	4	1001	-1	1110	-6
				1111	-7

Con *n* bits se pueden representar los números enteros en el rango [-2^{n-1} - 1, 2^{n-1} - 1]. (De -7_{10} a 7_{10}, 15 números)

Representación de nros. enteros binarios complemento a dos

El primer dígito indica el signo. Para positivos, el resto de los dígitos indica su magnitud. Para negativos, el resto de los dígitos representan el complemento de su magnitud +1.
Ej.: $7_{10} = 0111_{2,}$, $-7_{10} = compl_2\ (0111_2) + 1_2 = 1000_2 + 1_2 = 1001_2$

Ej.: Números binarios de 4 bits

Binario	Decimal	Binario	Decimal	Binario	Decimal
0000	0	0101	5	1010	-6
0001	1	0110	6	1011	-5
0010	2	0111	7	1100	-4
0011	3	1000	-8	1101	-3
0100	4	1001	-7	1110	-2
				1111	-1

Con *n* bits se pueden representar los números enteros en el rango $[-2^{n-1}, 2^{n-1}-1]$. (De -8_{10} a 7_{10})

Representación de números enteros en exceso Z

- La representación binaria de un número en exceso se obtiene sumando Z (2^{n-1} ó $2^{n-1}-1$) al número decimal y transformando el resultado a binario.
- Por ejemplo: La representación en exceso 32 de 16 es 48, de –22 es 10, de –16 es 16, de 0 es 32. La representación en exceso 8 de -8 es 0.
- De binario a decimal, a la representación del binario se le resta el exceso
- Representación en exceso 8:

Binario	Decimal	Binario	Decimal
0000	-8	1000	0
0001	-7	1001	1
0010	-6	1010	2
0011	-5	1011	3
0100	-4	1100	4
0101	-3	1101	5
0110	-2	1110	6
0111	-1	1111	7

El rango que se puede representar con *n* bits, en exceso 2^{n-1}, es $[-2^{n-1}, 2^{n-1}-1]$

Representación de números reales

No se puede establecer un máximo y un mínimo porque entre los dígitos están los dígitos decimales.

Nueva restricción: **Precisión**.

Las computadoras representan los números reales en notación científica normalizada, haciendo nula la parte entera.

Ej. $6789.213 \times 10^9 = 0.6789213 \times 10^{13}$

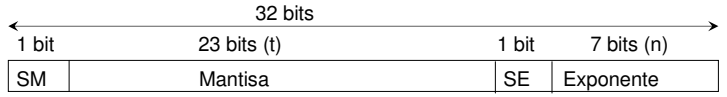
$100100.11 \times 2^7 = 0.10010011 \times 2^{13}$

Los números reales se pueden representar como: $\pm 0.m \times 10^e$ ó $\pm 0.m \times 2^e$ en binario, donde *m* se denomina mantisa y *e* exponente

m y *e* son valores enteros, pueden ser positivos o negativos.

Representación de números reales

Representación simple con 32 bits, rango $[2^{-127}, 2^{127}] \approx [10^{-38}, 10^{38}]$



$SM\ (0.a_1 a_2 a_3 \ldots a_t)_\beta \times \beta^{SE\ e}$

En dec. 1 a 9
En bin. 1

$a_1\beta^{-1} + a_2\beta^{-2} + \ldots + a_t\beta^{-t}$

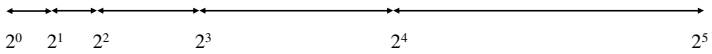
β : base
t : precisión
e : exponente, $0 < e < \beta^{n-1}$.

$$\overbrace{2}^{SM} * (\overbrace{\beta - 1}^{a_1}) * \overbrace{\beta^{t-1}}^{a_2 \ldots a_t} * \overbrace{2}^{SE} * \overbrace{\beta^{n-1}}^{e} \overbrace{+1}^{0}$$ números distintos

$2 * (2-1) * 2^{22} * 2 * (2^7-1) + 1 = 2{,}1307\ 10^9 = 2.130.706.433$

Precisión en números reales

$x = (-1)^{SM}. (0.1\ a_2...a_{22}). 2^{(-1)\ ^\wedge\ SE(ef - e0)}$



Rango del exponente [-127,127]
⇒ magnitud de números representables en binario desde
$0.1111...11 \times 2^{127} \approx 2^{127} \approx 10^{38}$ a $0.1000...0 \times 2^{-127} \approx 2^{-128} \approx 10^{-38}$

Pero la precisión no va a superar los 24 bits ⇒ precisión máxima es $2^{24} \approx 10^7$ o sea 7 dígitos decimales

Se puede evitar colocar el primer 1

Ejemplo

Mantisa$_2$	Exponente$_{10}$							
	-3 (001)	-2 (010)	-1 (011)	0 (100)	1 (101)	2 (110)	3 (111)	4 (000)
0.1000	0.0625	0.125	0.25	0.5	1	2	4	8
0.1001	0.070313	0.14063	0.28125	0.5625	1.125	2.25	4.5	9
0.1010	0.078125	0.15625	0.3125	0.625	1.25	2.5	5	10
0.1011	0.085938	0.17188	0.34375	0.6875	1.375	2.75	5.5	11
0.1100	0.09375	0.1875	0.375	0.75	1.5	3	6	12
0.1101	0.10156	0.20313	0.40625	0.8125	1.625	3.25	6.5	13
0.1110	0.10938	0.21875	0.4375	0.875	1.75	3.5	7	14
0.1111	0.11719	0.23438	0.46875	0.9375	1.875	3.75	7.5	15

Con una mantisa de 4 bits (3 bits al eliminar el primer 1) y un exponente de 3 bits (en representación exceso 4) podemos representar 64 números (sin considerar el signo), el resto de los números reales en el intervalo [0.0625,15] deberán ser aproximaciones.

La mantisa limita la precisión y el exponente el rango.

Conversión números reales decimales ↔ binarios

Binario a decimal

$1101.011_2 = 1*2^3 + 1*2^2 + 0*2^1 + 1*2^0 + 0*2^{-1} + 1*2^{-2} + 1*2^{-3} = 13.375_{10}$

Decimal a binario

- La parte entera se divide sucesivamente por 2 y se toman los restos.

```
13 | 2
 1  6 | 2
    0  3 | 2
       1  1
```

$13_{10} = 1101_2$

- Los decimales se obtienen multiplicando el número sucesivamente por 2 y extrayendo su parte entera.

0.375 * 2 = 0.750 **0**
0.750 * 2 = 1.50 **1**
0.500 * 2 = 1.00 **1**
0.000 * 2 = 0.00 **0**

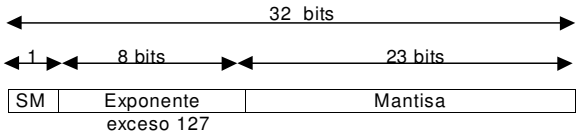
$0.375_{10} = 0.011_2$

$13.375_{10} = 13_{10} + 0.375_{10} = 1101.011_2$
luego de normalizado, $= 0.1101011_2 * 2^4$

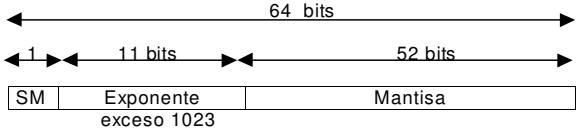
Representación IEEE

(Instituto de Ingeniería Eléctrica y Electrónica - 1985)

Representación IEEE754 con 32 bits



representación IEEE754 con 64 bits



Características de la representación IEEE

$x = \pm (1 + f)\, 2^{e}$ con $0 \le f < 1$

f determina precisión. $f = (\text{entero} < 2^{52}) / 2^{52}$

e determina rango $-1022 \le e \le 1023$

Números especiales
- (e = - 1023 c/mantisa 0) = ±0
- (e = -1023 c/mantisa ≠ 0) = $\pm (0 + f)\, 2^{e}$
- (e = 1024 c/mantisa 0) = ± inf
- (e = 1024 c/mantisa ≠ 0) = NaN

Algunos valores en Matlab

Los números en punto flotante tienen un espacio discreto, un máximo y un mínimo.
eps es la distancia de un número al siguiente número en punto flotante.

	Binary	Decimal
eps	2^(-52)	2.2204e-16
realmin	2^(-1022)	2.2251e-308
realmax	(2-eps)*2^1023	1.7977e+308

2^1023 = 8.9985e+307
2^1023 * 2 = Inf
2^ -53 = 1.1102e-016
2^ -52 * 2^ -1022 = 4.94...e-324
2^ -53 * 2^ -1022 = 0

2^1023 /0 =inf
0 / 0 = NaN
2^ -53 +1 = 1
2^ -52 * 2^-1022 / 0 = 0
2^ -53 * 2^-1022 *2 = 0

Ejemplos

Ejemplo redondeo:
x = 1; while 1+x > 1, x = x/2, end
El resultado será: 53 valores de *x*, donde las últimos dos son *eps* y *eps*/2.

Ejemplo overflow (exceso):
*x = 1; while x+x > x, x = 2*x, end*
El resultado será: 1024 valores de *x*, donde las últimos dos son 2^1023 (≈ realmax/2) e *Inf.*(infinito)

Ejemplo underflow (anulación):
x = 1; while x+x > x, x = x/2, end
El resultado será: 1075 valores de *x*, donde las últimos dos son *eps*realmin* y 0.

Error de redondeo

La representación de un número real se denomina *representación en coma flotante. A c*ada valor $x \in \Re$ se le asocia un número de máquina mediante una función *fl (x).*

La diferencia entre *x* y el número obtenido x_{fl} se denomina ***error de redondeo***

$fl(x) = x_{fl} = x + \delta$ $\quad |\delta| \le eps = precisión = 2^{-52}$

Ejemplo errores de representación

*t = 0.1; n = 1:10; e = n/10 - n*t*

e = 1.0e-015 * [0 0 -0.0555 0 0 -0.1110 -0.11100 0 0]

n/10 tiene un bit menos que n*0.1

Errores en las operaciones

Los errores en las operaciones son producidos por la acumulación de errores de redondeo y la anulación de dígitos de precisión.

Operación:
- Se realiza en un registro de longitud mayor al de memoria
- Se normaliza el resultado
- Se redondea el resultado
- Se almacena el resultado en memoria

$\Rightarrow$ El error se propaga en operaciones sucesivas

$$\hat{p} = p + \varepsilon_p , \quad \hat{q} = q + \varepsilon_q$$

$$\hat{p} + \hat{q} = p + \varepsilon_p + q + \varepsilon_q = p + q + (\varepsilon_p + \varepsilon_q)$$

$$\hat{p} * \hat{q} = (p + \varepsilon_p) * (q + \varepsilon_q) = p * q + p * \varepsilon_q + q * \varepsilon_p + \varepsilon_p * \varepsilon_q$$

$$\hat{p} / \hat{q} = (p + \varepsilon_p) / (q + \varepsilon_q) = (p * q - p * \varepsilon_q + q * \varepsilon_p - \varepsilon_p * \varepsilon_q) / (q^2 - \varepsilon_q^2)$$

(multiplicando numerador y denominador por $(q - \varepsilon_q)$)

Propagación del error

a.-Consideremos la sucesión: $x_0=1$, $x_1=1/3$, $x_n=13/3* x_{n-1}- 4/3* x_{n-2}$

b.- Puede comprobarse que $x_n = (1/3)^n = x_{n-1}/3$

```
xa(1) = 1.0;  xa(2) = 1/3;
xb(1) = 1.0;  xb(2) = 1/3;
e_abs(1) = 0;  e_abs(2) = 0;
e_rel(1) = 0;  e_rel(2) = 0;
for i = 3: 20
   xa(i) = (13/3) * xa(i-1) - (4/3) * xa(i-2);
   xb(i) = xb(i-1) * 1 / 3;
   e_abs(i) = xa(i) - xb(i);
   e_rel(i) = e_abs(i) / xb(i);
   [i ,xa(i), xb(i), e_abs(i), e_rel(i)]
end
```

I	xa	xb	e_abs	e_rel
3	0.11111111111111	0.11111111111111	-0.00000000000000	-0.00000000000000
4	0.03703703703704	0.03703703703704	-0.00000000000000	-0.00000000000002
5	0.01234567901234	0.01234567901235	-0.00000000000000	-0.00000000000026
6	0.00411522633744	0.00411522633745	-0.00000000000001	-0.00000000000308
7	0.00137174211243	0.00137174211248	-0.00000000000005	-0.00000000003697
8	0.00045724737062	0.00045724737083	-0.00000000000020	-0.00000000044359
9	0.00015241578946	0.00015241579028	-0.00000000000081	-0.00000000532314
10	0.00005080526018	0.00005080526343	-0.00000000000325	-0.00000006387770
11	0.00001693507483	0.00001693508781	-0.00000000001298	-0.00000076653237
12	0.00000564497734	0.00000564502927	-0.00000000005193	-0.00000919838841
13	0.00000188146872	0.00000188167642	-0.00000000020770	-0.00011038066094
14	0.00000062639467	0.00000062722547	-0.00000000083080	-0.00132456793126
15	0.00000020575195	0.00000020907516	-0.00000000332321	-0.01589481517509
16	0.00000005639888	0.00000006969172	-0.00000001329284	-0.19073778210109
17	-0.00000002994080	0.00000002323057	-0.00000005317138	-2.28885338521304
18	-0.00000020494198	0.00000000774352	-0.00000021268550	-27.46624062255646
19	-0.000000848161	0.000000002581	-0.000000850742	-329.594887470677
20	-0.00000340211	0.00000000086	-0.00000340297	-3955.13864964813

Pérdida de cifras significativas

Se produce al realizar substracciones entre números similares.
(***cancelación substractiva***)

Ejemplo 1:

x = 0.3456842643, y = 0.3456704522, x - y = 1.38121000000102e-005

Si se reduce a 5 dígitos

x' = 0.34568 , y' = 0.34567, x' – y' = 0.00001

Error relativo = (x – y – (x' – y')) / (x - y) ≈ 27%

Ejemplo 2:

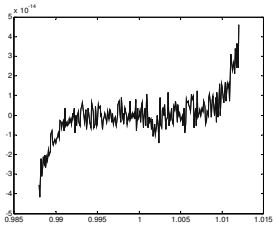
1e32 *(-100 + 100 + 1e-15)

ans = 1e+017

1e32 *(-100 + 1e-15 + 100) Alterando el orden de los sumandos

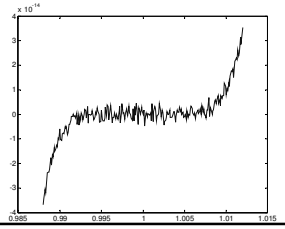
ans = 0

```
x = 0.989:0.0001:1.012
y = x.^7-7*x.^6+21*x.^5-35*x.^4+35*x.^3-21*x.^2+7*x-1;
plot(x,y)
```

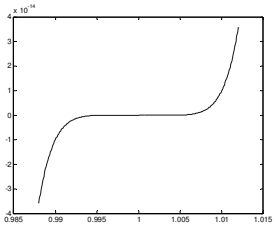


Cancelación substractiva

```
y=(((((((x-7).*x+21).*x-35).*x+35).*x-21).*x+7).*x-1;
plot(x,y)
```



```
y=(x-1).^7
plot(x,y)
```



Error de redondeo

$17x_1+5x_2=22$
$1.7x_1+0.5x_2=2.2$

$x_1=1$, $x_2=1$

```
>>A =
   17.0000    5.0000
    1.7000    0.5000

>> b=[22; 2.2]
b =
   22.0000
    2.2000
```

```
>> x=a\b

Warning: Matrix is close to singular or
badly scaled.
         Results may be inaccurate.
RCOND = 3.265362e-018.

x =
   -1.0588
    8.0000
```

La 2da ecuación es 0.1 de la primera, pero como el resultado no es exacto, no lo ve como una matriz singular

Algunos errores…:Antimisiles Patriot

El 25 de febrero de 1991, durante la guerra del Golfo, una bateríia de misiles Patriot americanos en Dharan (Arabia Saudí) no logró interceptar un misil Scud iraquí. Murieron 28 soldados americanos. La causa: los errores numéricos por utilizar truncado en lugar de redondeo en el sistema que calcula el momento exacto en que debe ser lanzado el misil.

Los ordenadores de los Patriot predicen la trayectoria del misil Scud, en función de su velocidad y del momento en que fue detectado por última vez en el radar.

La velocidad es un número real. El tiempo es una magnitud real, pero el sistema la calculaba mediante un reloj interno que contaba décimas de segundo, por lo que representaban el tiempo como una variable entera.

Los ordenadores del Patriot almacenan los números reales representados en punto flotante con una mantisa de 24 bits.

Para convertir el tiempo entero en un número real se multiplica éste por 1/10, y se trunca el resultado (en lugar de redondearlo).

La batería de los Patriot llevaba en funcionamiento más de100 horas, por lo que el tiempo entero era un número muy grande.

Veamos el cálculo en detalle.

La representación del número 0.1 en binario es
$(0.0001100110011001100110011001100...)_2$,
que almacenado en un registro de 24 bits conduce al número
$(0.00011001100110011001100)_2$
que introduce un error de $(0.0000000000000000000000011001100...)_2$,
igual en decimal a 0.000000095.

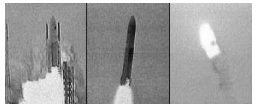
En 100 horas (transformando a décimas de segundo) este pequeño error se multiplica y amplifica hasta alcanzar
0.000000095 *100 * 60 * 60 * 10 = 0.34.
Como un misil Scud viaja a unos 1676 m/s, es decir, unos 6000 km/hora, en 0.34 segundos recorre más de medio kilómetro.

Esta distancia fue suficiente para que el misil Patriot no pudiera alcanzar al misil Scud y destruirlo.

http://www.cs.usyd.edu.au/~alum/patriot_bug.html

Algunos errores…: La Exploción del Ariane 5

El 4 de junio de 1996, el cohete Ariane 5, lanzado por la Agencia Espacial Europea explotó cuarenta segundos después su lanzamiento en Kourou, Guayana francesa.
El cohete estaba en su primer viaje, después de una década de desarrollo que costó u$s 7 billones. El cohete destruido y su carga estaban estimadas en u$s 500 millones.
El resultado de la investigación de las causas de la explosión resultó ser un error del software en el sistema de referencia inercial.
Específicamente un número en punto flotante de 64 bits, que relaciona la velocidad horizontal del cohete con respecto a la plataforma, se convirtió a un entero de 16 bits. El número era más grande que 32.767, y la conversión falló.



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

Fuentes de error en un modelo

- Datos (ruido, instrumentales)
- Variables consideradas
- Definición del modelo
- Representación de los números reales en la computadora
  - Naturaleza discreta (limitados)
  - Error de redondeo
  - Acumulación de errores
  - Error de desbordamiento

  } Precisión de cálculo

- Algoritmos de métodos Numéricos
  - Errores de aproximación
  - Algoritmos de cálculo inestables

  } Confiabilidad de los resultados

- Problemas mal condicionados

Conceptos básicos: Error (Incertidumbre)

Dado un valor *x* (generalmente desconocido) **y una aproximación $\underline{x}$**

Error Absoluto: Es la distancia entre dos valores $E_{abs} = |x - \underline{x}|$

$E_{abs} = |0.025 - 0.0256| = 0.0006$

$E_{abs} = |25000 - 25600| = 600$

Se expresa en las mismas unidades que la magnitud medida.

Error Relativo: Es una porción del valor exacto $E_{rel} = |x - \underline{x}| / |x|$

$E_{rel} = |0.025 - 0.0256| / |0.025| = 0.0240$

$E_{rel} = |25000 - 25600| / |25000| = 0.0240$

Expresa el error que se comete por cada unidad de la magnitud medida.

Cota de Error: Máximo valor del error. Es un $k \geq 0$ tal que $|E| \leq k$

Cifras Significativas

- Un dígito distinto de cero es significativo.
- Un cero es significativo si está ubicado entre dígitos distintos de cero.
- Los ceros a la izquierda del primer dígito diferente de cero no son significativos
- Si un número es mayor que 1, los ceros expresados a la derecha del punto decimal son cifras significativas.
- Si un número es menor que 1, sólo los ceros que están entre dígitos significativos son cifras significativas. Los ceros inmediatamente después del punto decimal no son significativos.

Ejemplos: Cantidad de cifras significativas

Número	C.Sig.
67489	5
3007	4
0,004	1
3,00	3
67,005	5
0,0000023	2
0,010001	5
0,0770	3

Número	C.Sig.	
500	3	*500* expresado con distinta cantidad de cifras significativas
$5x10^2$	1	
$5,0x10^2$	2	
$5,00x10^2$	3	
$67,005x10^4$	5	
$67,00x10^4$	4	Disminución de cifras significativas
$67x10^4$	2	
$67,000x10^4$	5	Pérdida de un dígito significativo al aumentar la cantidad de cifras significativas

Cifras decimales significativas

Al evaluar un error, se llaman cifras significativas a las que se consideran ciertas. Siendo $\underline{\boldsymbol{x}}$ una aproximación a $\boldsymbol{x}$, $\underline{\boldsymbol{x}}$ se aproxima a $\boldsymbol{x}$ con $\boldsymbol{d}$ Cifras Decimales Significativas (CDS), si d es el mayor entero no negativo tal que:

$|\boldsymbol{x} - \underline{\boldsymbol{x}}| < 10^{-d}$ ó $|\boldsymbol{x} - \underline{\boldsymbol{x}}| / |\boldsymbol{x}| < 10^{-d}$ (considerando corte) ó

$|\boldsymbol{x} - \underline{\boldsymbol{x}}| < 10^{-d}/2 = 5x10^{-(d+1)}$ ó $|\boldsymbol{x} - \underline{\boldsymbol{x}}| / |\boldsymbol{x}| < 10^{-d}/2 = 5x10^{-(d+1)}$ (por aproximación)

$E_{abs} = |0.025 - 0.0256| = 0.0006 < 10^{-3} = 0.001$ aprox con 3 CDS
$E_{abs} = |0.025 - 0.0256| = 0.0006 < 5x10^{-(2+1)} = 0.005$ aprox con 2 CDS
$E_{abs} = |0.025 - 0.0253| = 0.0003 < 5x10^{-(3+1)} = 0.0005$ aprox con 3 CDS
$E_{rel} = |0.025 - 0.0256| / |0.025| = 0.0240 < 10^{-1}$ ó $< 10^{-1}/2$

Si $\boldsymbol{d} \geq$ dígitos utilizados para la representación se produce la Pérdida de cifras decimales significativas o cancelación

Cálculo Simbólico

- Utiliza aritmética racional
- Produce resultados exactos
- Mayor costo computacional (tiempo y memoria)
- No siempre puede aplicarse (método analítico)

```
diff('sin(2*x+x)')
        ans =3*cos(3*x)
diff('x+2*y+x*y','y')
        ans =2+x
syms a b c x
S = a*x^2 + b*x + c;
solve(S)
        ans =
        [1/2/a*(-b+(b^2-4*a*c)^(1/2))]
        [1/2/a*(-b-(b^2-4*a*c)^(1/2))]
```

```
b = solve(S, b)
     b =  - (a*x^2+c)/x
s = solve('cos(2*x)+sin(x)=1')
     s =
     [ 0]
     [ pi]
     [ 1/6*pi]
     [ 5/6*pi]
```

Métodos iterativos

Generalmente los métodos numéricos utilizan procesos repetitivos para obtener un resultado, estos consisten en sustituir reiteradamente un valor por el resultado de aplicarle una fórmula o función .Ese proceso se denomina iteración

Se requiere de una fórmula o función *f(x)* y de un valor de partida x_0.

Valor de partida x_0

$x_1 = f(x_0)$
$x_2 = f(x_1) = f(f(x_0))$
$x_3 = f(x_2) = f(f(x_1)) = f(f(f(x_0)))$
...
$x_n = f(x_{n-1}) = f(f(x_{n-2})) = f(f(f(x_{n-3}))) = ... = f(f(.....(x_0)...)$

Aproximación por procesos iterativos

- La sucesión de valores $x_0, x_1, \ldots, x_n$, puede ser infinita, tendiendo o no a un valor de interés.
- Cuando la sucesión tiende a un valor de interés decimos que es *convergente*, pudiendo ser de importancia la velocidad con la cual converge (cantidad de iteraciones necesarias).
- La velocidad de convergencia se denomina *orden de convergencia* y depende del método.
- Si la sucesión tiende al valor de interés, limitar la cantidad de iteraciones puede llevar a un *error de truncamiento* de la sucesión.

Convergencia

Sea:

$\boldsymbol{r}$ la respuesta a un problema

$\boldsymbol{x_i}$ el valor calculado en la iteración $\boldsymbol{i}$

$\boldsymbol{x_{i+1}}$ el valor calculado en la iteración $\boldsymbol{i+1}$ con $\boldsymbol{h} = | \boldsymbol{x}_{i+1} - \boldsymbol{x}_i |$

- *Error de truncamiento*

$\boldsymbol{E_i} = |\boldsymbol{r} - \boldsymbol{x_i}|$ Error absoluto de truncamiento en la iteración $\boldsymbol{i}$

$\boldsymbol{E_{i+1}} = |\boldsymbol{r} - \boldsymbol{x_{i+1}}|$ Error absoluto de truncamiento en la iteración $\boldsymbol{i+1}$

- *Convergencia*

$\boldsymbol{x}_i \underset{i\to\infty}{\to} \boldsymbol{r}$ lo cual es equivalente a $\underset{i\to\infty}{\boldsymbol{lim}}\ \boldsymbol{E}_i = 0$

- *Orden de convergencia*

Si $\boldsymbol{x_i} \to \boldsymbol{r} \Rightarrow |\boldsymbol{E_{i+1}}| < |\boldsymbol{E_i}|$ Si la relación se puede expresar como:

$|\boldsymbol{E_{i+1}}| = k |\boldsymbol{E_i}|$ con $0<k<1 \Rightarrow$ la convergencia es lineal o de orden 1

Si $|\boldsymbol{E_{i+1}}| = k |\boldsymbol{E_i}|^n \Rightarrow$ la convergencia es de orden n

Y se expresa como $O(h^n)$

Algoritmos Inestables

Un algoritmo es inestable cuando los errores de redondeo se acumulan degradando el resultado final.

Un problema está mal condicionado si la solución es muy sensible a pequeños cambios en las variables de entrada.

Al resolver un problema en un computador introducimos pequeños cambios en las variables de entrada por el cambio de la representación.

El mal condicionamiento es propio de un problema, pero se puede reflejar al resolverlo por medio de una computadora