1. Introducción

La ciencia y la tecnología describen los fenómenos reales mediante modelos matemáticos. El estudio de estos modelos permite un conocimiento más profundo del fenómeno, así como de su evolución futura. La matemática aplicada es la rama de las matemáticas que se dedica a buscar y aplicar las herramientas más adecuadas a los problemas basados en estos modelos. Desafortunadamente, no siempre es posible aplicar métodos analíticos clásicos por diferentes razones:

- No se adecúan al modelo concreto.
- Su aplicación resulta excesivamente compleja.
- La solución formal es tan complicada que hace imposible cualquier interpretación posterior.
- Simplemente no existen métodos analíticos capaces de proporcionar soluciones al problema.

En estos casos son útiles las técnicas numéricas, que mediante una labor de cálculo más o menos intensa, conducen a soluciones aproximadas que son siempre numérica. El importante esfuerzo de cálculo que implica la mayoría de estos métodos hace que su uso esté íntimamente ligado al empleo de computadores. De hecho, sin el desarrollo que se ha producido en el campo de la informática resultaría difícilmente imaginable el nivel actual de utilización de las técnicas numéricas en ámbitos cada día más diversos[1].

---

2. Errores

El concepto de error es consustancial con el cálculo numérico. En todos los problemas es fundamental hacer un seguimiento de los errores cometidos a fin de poder estimar el grado de aproximación de la solución que se obtiene.

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

- Aquellos que son inherentes a la formulación del problema.
- Los que son consecuencia del método empleado para encontrar la solución del problema.

Dentro del grupo de los primeros, se incluyen aquellos en los que la definición matemática del problema es sólo una aproximación a la situación física real. Estos errores son normalmente despreciables; por ejemplo, el que se comete al obviar los efectos relativistas

en la solución de un problema de mecánica clásica. En aquellos casos en que estos errores no son realmente despreciables, nuestra solución será poco precisa independientemente de la precisión empleada para encontrar las soluciones numéricas.

Otra fuente de este tipo de errores tiene su origen en la imprecisión de los datos físicos: constantes físicas y datos empíricos. En el caso de errores en la medida de los datos empíricos y teniendo en cuenta su carácter generalmente aleatorio, su tratamiento analítico es especialmente complejo pero imprescindible para contrastar el resultado obtenido computacional-mente.

En lo que se refiere al segundo tipo de error (error computacional), tres son sus fuentes principales:

1. Equivocaciones en la realización de las operaciones (errores de bulto). Esta fuente de error es bien conocida por cualquiera que haya realizado cálculos manualmente o empleando una calculadora. El empleo de computadores ha reducido enormemente la probabilidad de que este tipo de errores se produzcan. Sin embargo, no es despreciable la probabilidad de que el programador cometa uno de estos errores (calculando correctamente el resultado erróneo). Más aún, la presencia de *bugs* no detectados en el compilador o en el software del sistema no es inusual. Cuando no resulta posible verificar que la solución calculada es razonablemente correcta, la probabilidad de que se haya cometido un error de bulto no puede ser ignorada. Sin embargo, no es esta la fuente de error que más nos va a preocupar.
2. El error causado por resolver el problema no como se ha formulado, sino mediante algún tipo de aproximación. Generalmente está causado por la sustitución de un *infinito* (sumatorio o integración) o un *infinitesimal* (diferenciación) por una aproximación finita. Algunos ejemplos son:

    - El cálculo de una función elemental (por ejemplo, Seno x) empleando sólo n términos de los infinitos que constituyen la expansión en serie de Taylor.
    - Aproximación de la integral de una función por una suma finita de los valores de la función, como la empleada en la regla del trapezoide.
    - Resolución de una ecuación diferencial reemplazando las derivadas por una aproximación (diferencias finitas).
    - Solución de la ecuación $f(x) = 0$ por el método de Newton-Raphson: proceso iterativo que, en general, converge sólo cuando el número de iteraciones tiende a infinito.

    Denominaremos a este error, en todas sus formas, como *error por truncamiento*, ya que resulta de truncar un proceso infinito para obtener un proceso finito. Obviamente, estamos interesados en estimar, o al menos acotar, este error en cualquier procedimiento numérico.
3. Por último, la otra fuente de error de importancia es aquella que tiene su origen en el hecho de que los cálculos aritméticos no pueden realizarse con precisión

ilimitada. Muchos números requieren infinitos decimales para ser representados correctamente, sin embargo, para operar con ellos es necesario redondearlos. Incluso en el caso en que un número pueda representarse exactamente, algunas operaciones aritméticas pueden dar lugar a la aparición de errores (las divisiones pueden producir números que deben ser redondeados y las multiplicaciones dar lugar a más dígitos de los que se pueden almacenar). El error que se introduce al redondear un número se denomina *error de redondeo*.

2.1 Definiciones

Ahora que disponemos de una idea correcta de qué es el error y de cual es su origen, podemos formalizar el concepto de error. Generalmente, no conocemos el valor de una cierta magnitud $\bar{x}$ y hemos de conformarnos con un valor aproximado *x*. Para estimar la magnitud de este error necesitamos dos definiciones básicas:

Error absoluto
de *x*:

$$e_a(x) = x - \bar{x} \tag{1}$$

Error relativo
de *x*:

$$e_r(x) = \frac{e_a(x)}{\bar{x}}; \qquad (\bar{x} \neq 0) \tag{2}$$

En la práctica, se emplea la expresión:

$$e_r(x) = \frac{e_a(x)}{x}; \qquad (x \neq 0) \tag{3}$$

En general, no conocemos el valor de este error, ya que no es habitual disponer del valor exacto de la magnitud, sino sólo de una acotación de su valor, esto es, un número $\varepsilon(x)$, tal que:

$$|e_a(x)| \leq \varepsilon_a(x) \tag{4}$$

o bien:

$$|e_r(x)| \leq \varepsilon_r(x) \tag{5}$$

De acuerdo con este formalismo, tenemos que un numero se representará del siguiente modo:

$$\bar{x} = x \pm \varepsilon_a(x) \tag{6}$$

$$\bar{x} = x(1 \pm \varepsilon_r(x)) \tag{7}$$

2.2 Dígitos significativos

Sea x un número real que, en general, tiene una representación decimal infinita. Podemos decir que x ha sido adecuadamente redondeado a un número con d decimales, al que denominaremos $x^{(d)}$, si el error de redondeo, ε es tal que:

$$|\varepsilon_r^{(d)}| = |x - x^d| \leq \frac{1}{2} \cdot 10^{-d} \tag{8}$$

Ejemplo 1: Exprese el número x=35.47846 correctamente redondeado a cuatro ($x^{(4)}$) y tres ($x^{(3)}$) decimales. Calcular el error cometido.

Solución: en el primer caso obtenemos:

$x^{(4)} = 35.4785$

$$\varepsilon_r^{(4)} = |35.47846 - 35.4785| = 4.0 \cdot 10^{-5} \leq \frac{1}{2} \cdot 10^{-4}$$

En el segundo caso, la aproximación correcta es:

$x^{(3)} = 35.478$

$$\varepsilon_r^{(4)} = |35.47846 - 35.478| = 4.6 \cdot 10^{-4} \leq \frac{1}{2} \cdot 10^{-3}$$

y no la siguiente:

$x^{(3)} = 35.479$

$$\varepsilon_r^{(3)} = |35.47846 - 35.479| = 5.4 \cdot 10^{-4} \geq \frac{1}{2} \cdot 10^{-3}$$

Es decir, no es correcto redondear por exceso cuando el dígito anterior es 5 y proviene de un acarreo previo.

Otra forma de obtener el número de cifras significativas es mediante *truncamiento*, en donde simplemente se eliminan los dígitos de orden inferior. El error cometido en este caso es:

$$|\varepsilon_t^{(d)}| = |x - x^d| \leq 1 \cdot 10^{-d} \tag{9}$$

y que, en general, conduce a peores resultados que el método anterior.

Ejemplo 2: Exprese el número x=35.47846 truncado a cuatro ($x^{(4)}$) y tres ($x^{(3)}$) decimales. Calcular el error cometido.

Solución:

$x^{(4)} = 35.4784$

$$\varepsilon_t^{(4)} = |35.47846 - 35.4784| = 6.0 \cdot 10^{-5} \leq 1 \cdot 10^{-4}$$

$x^{(3)} = 35.478$

$$\varepsilon_r^{(4)} = |35.47846 - 35.478| = 4.6 \cdot 10^{-4} \leq 1 \cdot 10^{-3}$$

2.3 Propagación de errores

Cuando se resuelve un problema matemático por métodos numéricos y aunque las operaciones se lleven a cabo exactamente, obtenemos una aproximación numérica del

resultado exacto. Es importante tratar de conocer el efecto que sobre el resultado final del problema tiene cada una de las operaciones realizadas.

Para estudiar como se propaga en error, veamos cual es el efecto que cada una de las operaciones básicas tiene sobre el error final cuando se aplican sobre dos números $x_1 \pm \varepsilon_a(x_1)$ y $x_2 \pm \varepsilon_a(x_2)$:

$$\varepsilon_a(x_1 + x_2) = \varepsilon_a(x_1) + \varepsilon_a(x_2) \quad (10)$$

$$\varepsilon_a(x_1 - x_2) = \varepsilon_a(x_1) + \varepsilon_a(x_2) \quad (11)$$

$$\varepsilon_r(x_1 x_2) = \varepsilon_r(x_1) + \varepsilon_r(x_2) \quad (12)$$

$$\varepsilon_r\left(\frac{x_1}{x_2}\right) = \varepsilon_r(x_1) + \varepsilon_r(x_2) \quad (13)$$

Cuando el problema consiste en calcular el resultado $y = f(x)$tenemos la siguiente fórmula aproximada de propagación del error:

$$\varepsilon_a(y) = |f'(x)|\varepsilon_a(x) \quad (14)$$

En el caso más general, en que una función depende de más de una variable ($y = f(x_1, x_2, \ldots, x_n)$), la fórmula aproximada de propagación del error maximal es:

$$\varepsilon_a(y) = \sum_{i=1}^{n} \left|\frac{\partial}{\partial x_i} f(x_1, x_2, \ldots, x_n)\right| \varepsilon_a(x_i) \quad (15)$$

Ejemplo 3: Determinar el error máximo cometido en el cálculo $y = x_1 x_2^2$ para $x_1 = 2.0 \pm 0.1$ y $x_2 = 3.0 \pm 0.2$.

Solución: El error cometido, de acuerdo con la ecuación (15), se puede calcular mediante:

$$\begin{aligned}\varepsilon_a(y) &= \left|\frac{\partial}{\partial x_1}(x_1x_2^2)\right|\varepsilon_a(x_1) + \left|\frac{\partial}{\partial x_2}(x_1x_2^2)\right|\varepsilon_a(x_2)\\ \varepsilon_a(y) &= |x_2^2|\varepsilon_a(x_1) + |2x_1x_2|\varepsilon_a(x_2)\end{aligned}$$

Sustituyendo valores, obtenemos:

$$\begin{aligned}y &= 2.0\cdot(3.0)^2 = 18.00\\ \varepsilon_a(y) &= (3.0)^2\cdot 0.1 + 2\cdot 2.0\cdot 3.0\cdot 0.2 = 3.3\end{aligned}$$

Por lo que el resultado final se debe expresar como:

$$\boxed{y = 18 \pm 3}$$

Ejemplo 4: Sea el siguiente sistema de ecuaciones lineales:

$$\left.\begin{aligned}x + ay &= 5\\ bx + 2y &= d\end{aligned}\right\}$$

en donde $a = 1.000 \pm 0.002$; $b = 1 / a$ y $d = b - a$¿Con qué exactitud podemos determinar el producto xy?

Solución: Primero resolveremos el sistema de ecuaciones por reducción:

$$\begin{aligned}x &= \frac{(10 - ad)}{(2 - ab)}\\ y &= \frac{(d - 5b)}{(2 - ab)}\end{aligned}$$

Ecuaciones que conducen a la siguiente expresión para el producto:

$$xy = \frac{\overbrace{(10 - ad)}^{A}\overbrace{(d - 5b)}^{B}}{\underbrace{(2 - ab)^2}_{C}} \qquad (16)$$

Resolveremos ahora el problema por dos métodos. Primero, calcularemos el error asociado a cada una de las variables y los términos de la expresión anterior:

$$
\begin{array}{ll}
a = 1.000; & \varepsilon_a(a) = 0.002 \\
b = \frac{1}{a} = 1; & \varepsilon_a(b) = \frac{\varepsilon_a(a)}{a^2} = \varepsilon_a(a) \\
d = b - a = 0; & \varepsilon_a(d) = \varepsilon_a(b) + \varepsilon_a(a) = 2\varepsilon_a(a) \\
A = (10 - ad) = 10; & \varepsilon_a(A) = a\varepsilon_a(d) + d\varepsilon_a(a) = 2\varepsilon_a(a) \\
B = (d - 5b) = -5; & \varepsilon_a(B) = \varepsilon_a(d) + 5\varepsilon_a(b) = 7\varepsilon_a(a) \\
C = (2 - ab)^2 = 1; & \varepsilon_a(C) = 2\varepsilon_a(a) + 2\varepsilon_a(b) + 4\varepsilon_a(b) + 4\varepsilon_a(a) = 12\varepsilon_a(a) \\
AB = -50; & \varepsilon_a(AB) = B\varepsilon_a(A) + A\varepsilon_a(B) = 80\varepsilon_a(a) \\
xy = \frac{AB}{C} = -50; & \varepsilon_a(xy) = \frac{\varepsilon_a(AB)}{C} + \frac{AB\varepsilon_a(C)}{C^2} = 680\varepsilon_a(a)
\end{array}
$$

Sustituyendo valores, obtenemos el siguiente resultado:

$$\boxed{xy = -50.0 \pm 1.4}$$

Una forma mucho más adecuada de resolver este problema consiste en sustituir en la expresión (16) los valores de *b* y *d* por sus correspondientes expresiones en función de *a*. Sustituyendo y operando, obtenemos que el producto y el error asociado vienen dados por:

$$
\begin{aligned}
xy &= -(a^2 + 13a + \frac{36}{a}) \\
\varepsilon_a(xy) &= (|2a| + 13 + |\frac{36}{a^2}|)\varepsilon_a(a)
\end{aligned}
$$

que, sustituyendo valores, conduce al resultado:

$$\boxed{xy = -50.00 \pm 0.10}$$

Si ambos resultados son correctos ¿Por qué el error es mucho menor en el segundo caso que en el primero? La respuesta es simple: en el segundo caso hemos eliminado operaciones intermedias, permitiendo que algunos errores se cancelen mutuamente. En general, cuanto menor sea el número de pasos intermedios que efectuemos para alcanzar la solución, menor será el error cometido.

2.4 Ejercicios adicionales

1.

¿Con qué exactitud es necesario medir el radio de una esfera para que su volumen sea conocido con un error relativo menor de 0.01%? ¿Cuantos decimales es necesario emplear para el valor de π?

Soluciones: $\varepsilon_r(R) \leq \frac{1}{3} \cdot 10^{-4}$. El número π debe expresarse al menos con seis cifras decimales.

2.

Supongamos una barra de hierro de longitud l y sección rectangular $a \times b$ fija por uno de sus extremos. Si sobre el extremo libre aplicamos una fuerza F perpendicular a la barra, la flexión s que ésta experimenta viene dada por la expresión:

$$s = \frac{4}{E}\frac{l^3}{ab^3}F$$

en donde E es una constante que depende sólo del material denominada módulo de Young. Conociendo que una fuerza de 140 Kp aplicada sobre una barra de 125 cm de longitud y sección cuadrada de 2.5 cm produce una flexión de 1.71 mm, calcular el módulo de Young y el intervalo de error. Suponer que los datos vienen afectados por un error máximo correspondiente al de aproximar por truncamiento las cifras dadas.

3. Aritmética de computadores

Los computadores no almacenan los números con precisión infinita sino de forma aproximada empleando un número fijo de *bits* (apócope del término inglés *Binary Digit*) o *bytes* (grupos de ocho *bits*). Prácticamente todos los computadores permiten al programador elegir entre varias representaciones o 'tipos de datos'. Los diferentes tipos de datos pueden diferir en el número de bits empleados, pero también (lo que es más importante) en cómo el número representado es almacenado: en formato fijo (también denominado 'entero') o en punto flotante[2] (denominado 'real').

.1 Aritmética de punto fijo

Un entero se puede representar empleando todos los bits de una palabra de computadora, con la salvedad de que se debe reservar un bit para el signo. Por ejemplo, en una máquina con longitud de palabra de 32 bits, los enteros están comprendidos entre $-(2^{31} - 1)$ y $2^{31} - 1 = 2147483647$. Un número representado en formato entero es 'exacto'. Las operaciones aritméticas entre números enteros son también 'exactas' siempre y cuando:

1. La solución no esté fuera del rango del número entero más grande o más pequeño que se puede representar (generalmente con signo). En estos casos se dice que se comete un error de desbordamiento por exceso o por defecto (en inglés: *Overflow* y *Underflow*) y es necesario recurrir a técnicas de escalado para llevar a cabo las operaciones.
2.

La división se interpreta que da lugar a un número entero, despreciando cualquier resto.

Por estos motivos, la aritmética de punto fijo se emplea muy raramente en cálculos no triviales.

3.2 Números en punto flotante

3.2.1 Notación científica normalizada

En el sistema decimal, cualquier número real puede expresarse mediante la denominada ***notación científica normalizada***. Para expresar un número en notación científica normalizada multiplicamos o dividimos por 10 tantas veces como sea necesario para que todos los dígitos aparezcan a la derecha del punto decimal y de modo que el primer dígito después del punto no sea cero. Por ejemplo:

$$\begin{aligned} 732.5051 &= 0.7325051 \times 10^3 \\ -0.005612 &= -0.5612 \times 10^{-2} \end{aligned}$$

En general, un número real *x* distinto de cero, se representa en notación científica normalizada en la forma:

$$x = \pm r \times 10^n \tag{17}$$

en donde *r* es un número tal que $\frac{1}{10} \leq r < 1$ y *n* es un entero (positivo, negativo o cero).

Exactamente del mismo modo podemos utilizar la notación científica en el sistema *binario*. En este caso, tenemos que:

$$x = \pm q \times 2^m \tag{18}$$

donde *m* es un entero. El número *q* se denomina **mantisa** y el entero *m* **exponente**. En un ordenador binario tanto *q* como *m* estarán representados como números en base 2. Puesto que la mantisa *q* está normalizada, en la representación binaria empleada se cumplirá que:

$$\frac{1}{2} \leq |q| < 1 \tag{19}$$

3.2.2 Representación de los números en punto flotante

En un ordenador típico los números en punto flotante se representan de la manera descrita en el apartado anterior, pero con ciertas restricciones sobre el número de dígitos de q y m impuestas por la longitud de palabra disponible (es decir, el número de *bits* que se van a emplear para almacenar un número). Para ilustrar este punto, consideraremos un ordenador hipotético que denominaremos `MARC-32` y que dispone de una longitud de palabra de 32 *bits* (muy similar a la de muchos ordenadores actuales). Para representar un número en punto flotante en el `MARC-32`, los *bits* se acomodan del siguiente modo:

Signo del número real x:	1 bit
Signo del exponente m:	1 bit
Exponente (entero $\|m\|$):	7 bits
Mantisa (número real $\|q\|$):	23 bits

En la mayoría de los cálculos en punto flotante las mantisas se normalizan, es decir, se toman de forma que el bit más significativo (el primer bit) sea siempre '1'. Por lo tanto, la mantisa q cumple siempre la ecuación (19).

Dado que la mantisa siempre se representa normalizada, el primer bit en q es siempre 1, por lo que no es necesario almacenarlo proporcionando un bit significativo adicional. Esta forma de almacenar un número en punto flotante se conoce con el nombre de ***técnica del 'bit fantasma'***.

Se dice que un número real expresado como aparece en la ecuación (18) y que satisface la ecuación (19) tiene la forma de ***punto flotante normalizado***. Si además puede representarse exactamente con $|m|$ ocupando 7 bits y $|q|$ ocupando 24 bits, entonces es un ***número de máquina*** en el `MARC-32`[3]

La restricción de que $|m|$ no requiera más de 7 bits significa que:

$$|m| \leq (1111111)_2 = 2^7 - 1 = 127$$

Ya que $2^{127} \approx 10^{38}$, la `MARC-32` puede manejar números tan pequeños como 10^{-38} y tan grandes como 10^{38}. Este no es un intervalo de valores suficientemente generoso, por lo que en muchos casos debemos recurrir a programas escritos en ***aritmética de doble precisión*** e incluso de ***precisión extendida***.

Como q debe representarse empleando no más de 24 bits significa que nuestros números de máquina tienen una precisión limitada cercana a las siete cifras decimales, ya que el bit

menos significativo de la mantisa representa unidades de $2^{-24} \approx 10^{-7}$. Por tanto, los números expresados mediante más de siete dígitos decimales serán objeto de *aproximación* cuando se almacenen en el ordenador.

Por ejemplo: 0.5 representado en punto flotante en el `MARC-32` (longitud de palabra de 32 bits) se almacena en la memoria del siguiente modo:

$$\frac{1}{2} = \overbrace{0}^{\text{Signo } |m|} \quad \underbrace{0}_{\text{Signo } |q|} \quad \overbrace{0000000}^{|m|,\ 7 \text{ bits}} \underbrace{00000000000000000000}_{|q|,\ 1+23 \text{ bits}}$$

Ejemplo 5: Suponga un ordenador cuya notación de punto fijo consiste en palabras de longitud 32 bits repartidas del siguiente modo: 1 bit para el signo, 15 bits para la parte entera y 16 bits para la parte fraccionaria. Represente los números 26.32, $1.234 \cdot 10^{-4}$ y 12542.29301 en base 2 empleando esta notación de punto fijo y notación de punto flotante `MARC-32` con 32 bits. Calcule el error de almacenamiento cometido en cada caso.

Solución: El número 26.32 en binario se escribe del siguiente modo:

$$26.32_{10} = 11010.\overline{0101000111101011000}_2$$

Empleando las representaciones comentadas, obtenemos:

$$\begin{array}{rll} 26.32_{10} = & \textbf{Punto fijo :} & 0\ 000000000011010.0101000111101011 \\ & \textbf{Punto flotante :} & 0\ 10000101\ 10100101000111101011100 \end{array}$$

Si expresamos el error como la diferencia entre el valor y el número realmente almacenado en el ordenador, obtenemos:

$$\begin{array}{rcrcl} \varepsilon_a(\text{fix}) & = & 8 \cdot 10^{-6} & \varepsilon_r(\text{fix}) & = & 3 \cdot 10^{-7} \\ \varepsilon_a(\text{flt}) & = & 1.3 \cdot 10^{-6} & \varepsilon_r(\text{flt}) & = & 1.2 \cdot 10^{-8} \end{array}$$

En cuanto a los otros dos números, obtenemos:

$$\begin{array}{rll} 1.234 \cdot 10^{-6} = & 0\ 000000000000000.0000000000001000; & \varepsilon_r(\text{fix}) = 1.1 \cdot 10^{-2} \\ & 0\ 01110100\ 10000010110010011 10111; & \varepsilon_r(\text{flt}) = 1.7 \cdot 10^{-7} \\ 12542.29301 = & 0\ 0110000111111 10.0100101100000010; & \varepsilon_r(\text{fix}) = 9 \cdot 10^{-10} \\ & 0\ 10001110\ 11000011111110010010110; & \varepsilon_r(\text{flt}) = 3 \cdot 10^{-9} \end{array}$$

Antes de entrar con detalle en la aritmética de los números en punto flotante, es interesante notar una propiedad de estos números de especial importancia en los cálculos numéricos y que hace referencia a su densidad en la línea real. Supongamos que p, el número de bits de la mantisa, sea 24. En el intervalo $\left[\frac{1}{2}, 1\right)$ (exponente $f = 0$) es posible representar 2^{24} números igualmente espaciados y separados por una distancia $1/2^{24}$. De modo análogo, en cualquier intervalo $\left[2^f, 2^{f+1}\right)$ hay 2^{24} números equiespaciados, pero su densidad en este caso es $2^f/2^{24}$. Por ejemplo, entre $2^{20} = 1048576$ y $2^{21} = 2097152$ hay $2^{24} = 16777216$ números, pero el espaciado entre dos números sucesivos es de sólo $\frac{1}{16}$. De este hecho se deriva inmediatamente una regla práctica: cuando es necesario comparar dos números en punto flotante relativamente grandes, es siempre preferible comparar la diferencia relativa a la magnitud de los números. En la figura (1) se representa gráficamente la separación entre dos números consecutivos en función del exponente f en el rango $f = [20,30]$.

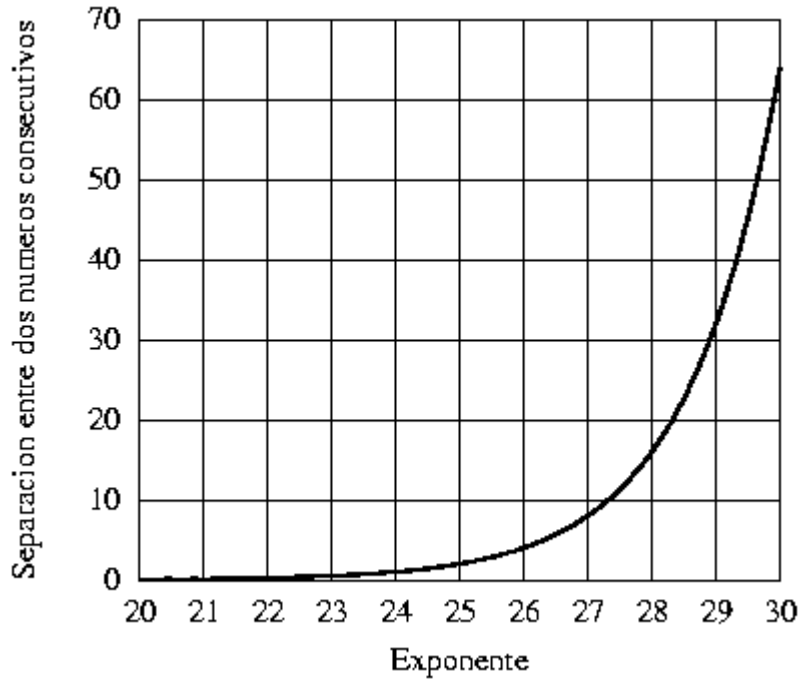

Figure: Evolución de la separación entre dos números consecutivos en función del exponente, f, de la representación en punto flotante de un número real.

[bb=55 60 455 410, clip=true, scale=0.7]eps/expon

3.3 Aritmética de punto flotante

En este apartado analizaremos los errores inherentes a la aritmética de los números de punto flotante. Primero consideraremos el error que surge como consecuencia de que los números reales no se pueden almacenar, en general, de forma exacta en un ordenador. Después analizaremos las cuatro operaciones aritméticas básicas y finalmente ampliaremos el estudio a un cálculo más complejo.

3.3.1 Números de máquina aproximados

Estamos interesados en estimar el error en que se incurre al aproximar un número real positivo *x* mediante un número de máquina del `MARC-32`. Si representamos el número mediante:

$$x = (a_1 a_2 \cdots a_{24} a_{25} a_{26} \cdots)_2 \times 2^m$$

en donde cada a_i es 0 ó 1 y el bit principal es $a_1 = 1$. Un número de máquina se puede obtener de dos formas:

- **Truncamiento**: descartando todos los bits excedentes $a_{25} a_{26} \cdots$. El número resultante, *x*' es siempre menor que *x* (se encuentra a la izquierda de *x* en la recta real).
- **Redondeo por exceso**: Aumentamos en una unidad el último bit remanente a_{24} y después eliminamos el exceso de bits como en el caso anterior.

Todo lo anterior, aplicado al caso del `MARC-32`, se resume diciendo que si *x* es un número real distinto de 0 dentro del intervalo de la máquina, entonces el número de máquina x^* más cercano a *x* satisface la desigualdad:

$$\delta = \left| \frac{x - x^*}{x} \right| \leq 2^{-24} \tag{20}$$

que se puede escribir de la siguiente forma:

$$x^* = x(1 + \delta) \qquad\qquad |\delta| \leq 2^{-24}$$

Ejemplo 6: ¿Cómo se expresa en binario el número *x* = 2/3? ¿Cuáles son los números de máquina *x*' y *x*'' próximos en el `MARC-32`?

El número 2/3 en binario se expresa como:

$$\left(\frac{2}{3}\right)_{10} = (0.\overline{10})_2$$

Los dos números de máquina próximos, cada uno con 24 bits, son:

$$x' = (0.101010\cdots1010)_2$$

$$x'' = (0.101010\cdots1011)_2$$

en donde x' se ha obtenido por truncamiento y x'' mediante redondeo por exceso.
Calculamos ahora las diferencias $x - x'$ y $x'' - x$ para estimar cual es el error cometido:

$$x - x' = \frac{2}{3} \times 10^{-24}$$

$$x'' - x = \frac{1}{3} \times 10^{-24}$$

Por tanto, el número más próximo es $fl(x) = x''$ y los errores de redondeo absoluto y relativo son:

$$|fl(x) - x| = \frac{1}{3} \times 10^{-24}$$

$$\left|\frac{fl(x) - x}{x}\right| = 2^{-25} < 2^{-24}$$

3.3.2 Las operaciones básicas

Vamos a analizar el resultado de operar sobre dos números en punto flotante normalizado de l-dígitos de longitud, x e y, que producen un resultado normalizado de l-dígitos. Expresaremos esta operación como:

$$fl(x\ op\ y)$$

en donde *op* es +, -, $\times$ ó $\div$. Supondremos que en cada caso la mantisa del resultado es primero normalizada y después redondeada (operación que puede dar lugar a un desbordamiento que requeriría renormalizar el número). El valor de la mantisa redondeada a p bits, q_r, se define como (de una forma más rigurosa que en el caso anterior):

$$q_r = \begin{cases} 2^{-p}\lfloor 2^p q + \frac{1}{2} \rfloor & q > 0 \\ 2^{-p}\lceil 2^p q - \frac{1}{2} \rceil & q < 0 \end{cases}$$

en donde la función *redondeo por defecto* $\lfloor x \rfloor$ es el mayor entero menor o igual a x y la función *redondeo por exceso* $\lceil x \rceil$ es el menor entero mayor o igual a x. Para números enteros, esta función se traduce en la bien conocida regla de sumar 1 en la posición $p + 1$. Teniendo en cuenta sólo la mantisa, redondear de este modo da lugar a un intervalo máximo del error de:

$$|\varepsilon_a| \leq 2^{-p-1} \tag{21}$$

y un error relativo máximo en el intervalo:

$$|\varepsilon_r| \leq \frac{2^{-p-1}}{1/2} = 2^{-p} \tag{22}$$

Analizaremos ahora el error generado por cada una de las operaciones básicas:

Multiplicación.

La operación de multiplicar dos números expresados en punto flotante implica sumar los exponentes y multiplicar las mantisas. Si la mantisa resultante no está normalizada, se recurre a renormalizar el resultado ajustando adecuadamente el exponente. Después, es necesario redondear la mantisa a p bits. Para analizar el error de esta operación supongamos dos números:

$$x = q_x 2^{f_x}; \quad y = q_y 2^{f_y}$$

Tenemos entonces que el producto será:

$$xy = q_x\, q_y\, 2^{f_x + f_y}$$

en donde el valor de la mantisa se encontrará en el rango:

$$\frac{1}{4} \leq |q_x q_y| < 1$$

ya que tanto *x* como *y* satisfacen la ecuación (19). Por tanto, la normalización del producto $q_x\, q_y$ implica un desplazamiento a la derecha de, como máximo, una posición. La mantisa redondeada será entonces uno de estos dos posibles valores:

$$x = q_x q_y + \varepsilon \quad \text{ó} \quad x = 2q_x q_y + \varepsilon$$

en donde ε, que es el error de redondeo, cumple la ecuación (21). Tenemos entonces:

$$\begin{aligned} fl(x \times y) &= \begin{cases} (q_x q_y + \varepsilon)2^{f_x+f_y} & |q_x q_y| \geq \frac{1}{2} \\ (2q_x q_y + \varepsilon)2^{f_x+f_y} & \frac{1}{2} > |q_x q_y| \geq \frac{1}{4} \end{cases} \\ &= q_x q_y 2^{f_x+f_y} \begin{cases} 1 + \frac{\varepsilon}{q_x q_y} & |q_x q_y| \geq \frac{1}{2} \\ 1 + \frac{\varepsilon}{2q_x q_y} & \frac{1}{2} > |q_x q_y| \geq \frac{1}{4} \end{cases} \\ &= xy(1 + \varepsilon_q) \end{aligned}$$

en donde, de acuerdo con la ecuación (22), tenemos:

$$|\varepsilon_q| \leq 2|\varepsilon| \leq 2^{-p}$$

Por tanto, la cota del error relativo en la multiplicación es la misma que la que surge por redondear la mantisa.

División.

Para llevar a cabo la división en punto flotante, se divide la mitad de la mantisa del numerador por la mantisa del denominador (para evitar cocientes mayores de la unidad), mientras que los exponentes se restan. Esto es:

$$\frac{x}{y} = \frac{q_x/2}{q_y} 2^{f_x - f_y + 1}$$

Puesto que ambas mantisas satisfacen la ecuación (18), el valor del cociente estará acotado entre los límites:

$$\frac{1}{4} \leq \left|\frac{q_x}{2q_y}\right| < 1$$

Aplicando un análisis similar al empleado en el caso de la multiplicación, obtenemos:

$$
\begin{aligned}
fl(x\ /\ y) &= \begin{cases} (q_x/2q_y+\varepsilon)2^{f_x-f_y+1} & |q_x/2q_y| \geq \frac{1}{2} \\ (q_x/q_y+\varepsilon)2^{f_x-f_y} & \frac{1}{2} > |q_x/2q_y| \geq \frac{1}{4} \end{cases} \\
&= q_x/2q_y 2^{f_x-f_y+1} \begin{cases} 1+\frac{\varepsilon}{q_x/2q_y} & |q_x/2q_y| \geq \frac{1}{2} \\ 1+\frac{\varepsilon}{q_x/q_y} & \frac{1}{2} > |q_x/2q_y| \geq \frac{1}{4} \end{cases} \\
&= \frac{x}{y}(1+\varepsilon_d)
\end{aligned}
$$

en donde, de acuerdo con la ecuación (22), tenemos:

$$|\varepsilon_d| \leq 2|\varepsilon| \leq 2^{-p}$$

Es decir, la cota máxima del error relativo en la división, como en el caso anterior, es la misma que la que surge por redondear la mantisa.

Adición y sustracción.

La operación de suma o resta se realiza del siguiente modo: se toma la mantisa del operando de menor magnitud (supongamos que es *y*) y se desplaza f_x - f_y posiciones a la derecha. La mantisa resultante es sumada (o restada) y el resultado se normaliza y después se redondea. Es decir:

$$x \pm y = (q_x \pm q_y 2^{f_y - f_x})2^{f_x}$$

El análisis del error cometido en esta operación es más complejo que los estudiados hasta ahora, por lo que no lo vamos a ver en detalle. Sin embargo, el resultado final indica que la cota máxima del error cometido en la adición y la sustracción viene dado por:

$$|\varepsilon_a| \leq 2|\varepsilon| \leq 2^{-p}$$

En conclusión, en todas las operaciones aritméticas elementales en punto flotante, el error absoluto del resultado es no mayor de 1 en el bit menos significativo de la mantisa.

Sin embargo, los errores de redondeo se acumulan a medida que aumenta el número de cálculos. Si en el proceso de calcular un valor se llevan a cabo *N* operaciones aritméticas es posible obtener, en el mejor de los casos, un error de redondeo total del orden de $\sqrt{N}\epsilon_m$ [4] (que coincide con el caso en que los errores de redondeo están aleatoriamente distribuidos, por lo que se produce una cancelación parcial). Desafortunadamente, este error puede crecer muy rápidamente por dos motivos:

- Es muy frecuente que la regularidad del cálculo o las peculiaridades del computador den lugar a que el error se acumule preferentemente en una dirección; en cuyo caso el error de redondeo se puede aproximar a $N\epsilon_m$.
- En circunstancias especialmente desfavorables pueden existir operaciones que incremente espectacularmente el error de redondeo. Generalmente, este fenómeno se da cuando se calcula la diferencia entre dos números muy próximos, dando lugar a un resultado en el cual los únicos bits significativos que no se cancelan son los de menor orden (en los únicos en que difieren). Puede parecer que la probabilidad de que se de dicha situación es pequeña, sin embargo, algunas expresiones matemáticas fomentan este fenómeno.

Veamos con un ejemplo los problemas comentados anteriormente. Hay dos formas de calcular las soluciones de la familiar ecuación cuadrática:

$$ax^2 + bx + c = 0$$

que son:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{23}$$

$$x = \frac{2c}{-b \pm \sqrt{b^2 - 4ac}} \tag{24}$$

Cualquiera de estas dos expresiones da problemas cuando a, c o ambos, son pequeños. En estos casos, el valor del discriminante es muy próximo al valor de b:

$$\sqrt{b^2 - 4ac} \approx b$$

$$(|b| - \sqrt{b^2 - 4ac})$$

por lo que la diferencia viene afectada de un error de redondeo importante. En efecto, la ecuación (23) evalúa bien la raíz más grande en valor absoluto, pero da pésimos resultados al estimar la raíz menor en valor absoluto. Por otra parte, la ecuación (24) calcula bien la raíz menor (siempre en valor absoluto) pero no la raíz más grande.

La solución del problema pasa por emplear una expresión mejor condicionada. En este caso, es preferible calcular previamente:

$$q = -\frac{1}{2}\left[b + \text{sign}(b)\sqrt{b^2 - 4ac}\right] \tag{25}$$

y las dos raíces a partir de valor de q como:

$$x_1 = \frac{q}{a} \quad \text{y} \quad x_2 = \frac{c}{q} \tag{26}$$

Ejemplo: Calcular las raíces de la siguiente ecuación cuadrática:

$$ax^2 + bx + c = 0$$

siendo:

$$a = 1.0; \quad b = 1.343 \cdot 10^5; \quad 3.764 \cdot 10^{-6}$$

Solución: Empleando la ecuación (23), obtenemos:

$$\begin{aligned} x_1 &= \underline{-0.13430 \cdot 10^5} \\ x_2 &= 0.44676 \cdot 10^{-3} \end{aligned}$$

Sin embargo, empleando la expresión (24):

$$\begin{aligned} x_1 &= \infty \quad (overflow) \\ x_2 &= \underline{-0.28027 \cdot 10^{-10}} \end{aligned}$$

Por último, empleando las expresiones (25) y (26) se obtienen ambas soluciones correctas:

$$\begin{aligned} x_1 &= \underline{-0.13430 \cdot 10^5} \\ x_2 &= \underline{-0.28027 \cdot 10^{-10}} \end{aligned}$$

3.4 Desbordamiento por exceso y desbordamiento por defecto

En la discusión anterior, hemos ignorado la posibilidad de que el resultado de una operación del punto flotante pueda no ser representable mediante el esquema fijo (l-bits)

empleado por el ordenador. La magnitud más grande que puede representarse mediante la fórmula general (18) es:

$$M \times 2^F \tag{27}$$

en donde F es el mayor exponente positivo representable (generalmente 2^7 - 1) y M es la mantisa que tiene todos sus bits puestos a 1 ($M = 1 - 2^{24}$). Un desbordamiento por exceso de punto flotante (*overflow* en inglés) se origina cuando el resultado de una operación de punto flotante tiene una magnitud mayor que la representada por la ecuación (27).

Ejemplo: Con q = 8 (y por tanto $F = 2^7 - 1 = 127$), las siguientes operaciones aritméticas dan lugar a desbordamiento por exceso:

$$\left(\tfrac{1}{2} \times 2^{70}\right) \times \left(\tfrac{1}{2} \times 2^{80}\right); \quad \left(\tfrac{1}{2} \times 2^{127}\right) - \left(-\tfrac{1}{2} \times 2^{127}\right)$$

El desbordamiento por defecto (*underflow* en inglés) se produce cuando el resultado de una operación en punto flotante es demasiado pequeño, aunque no nulo, como para que se pueda expresar en la forma dada por la ecuación (18). El número más pequeño representable suponiendo que siempre trabajamos con mantisas normalizadas es $\frac{1}{2} \times 2^{-F}$, en donde $-F$ es el exponente negativo más grande permitido (generalmente -2^{-q-1}). Por ejemplo, con q=8 resulta $-F$ = -128.

Ejemplo: Con q = 8 (y por tanto $-F$ = -128), la siguiente operación aritmética da lugar a desbordamiento por defecto:

$$\frac{\frac{1}{2} \times 2^{-80}}{\frac{1}{2} \times 2^{50}};$$

El desbordamiento por exceso es casi siempre resultado de un error en el cálculo. Sin embargo, en el caso del desobordamiento por defecto, en muchas ocasiones es posible continuar el cálculo reemplazando el resultado por cero.

3.5 Condicionamiento y estabilidad

La 'inestabilidad' en un cálculo es un fenómeno que se produce cuando los errores de redondeo individuales se propagan a través del cálculo incrementalmente. Veamos

brevemente este fenómeno y el problema relacionado con este: el 'condicionamiento' del método o del problema.

La mejor forma de ver este fenómeno es a través de un ejemplo. Supongamos el siguiente sistema de ecuaciones diferenciales:

$$\begin{aligned} y_1' &= y_2 \\ y_2' &= y_1 \end{aligned}$$

que tiene la siguiente solución general:

$$\begin{aligned} y_1 &= a_1 e^x + a_2 e^{-x} \\ y_2 &= a_1 e^x - a_2 e^{-x} \end{aligned}$$

En el caso particular en que las condiciones iniciales de nuestro problema son:

$$y_1(0) = -y_2(0) = 1$$

es posible determinar que el valor de las constantes a_1 y a_2 es: $a_1 = 0$ & y & $a_2 = 1$

Hasta este punto, las soluciones son exactas. Sin embargo, supongamos que el sistema de ecuaciones anterior se resuelve empleando un método numérico cualquiera con el fin de calcular los valores de las funciones y_1 y y_2 en una secuencia de puntos $x_1, x_2, \ldots, x_n$ y que el error del método da lugar a un valor de $a_1 \neq 0$. Ya que a_1 multiplica a un exponencial creciente cualquier valor, por pequeño que sea, de a_1 dará lugar a que el término e^x domine sobre el término e^{-x} para valores suficientemente grandes de x (ver figura (2)). La conclusión que se obtiene es que no es posible calcular una solución al sistema de ecuaciones diferenciales anterior que, para valores suficientemente grandes de x, no de lugar a un error arbitrariamente grande en relación con la solución exacta.

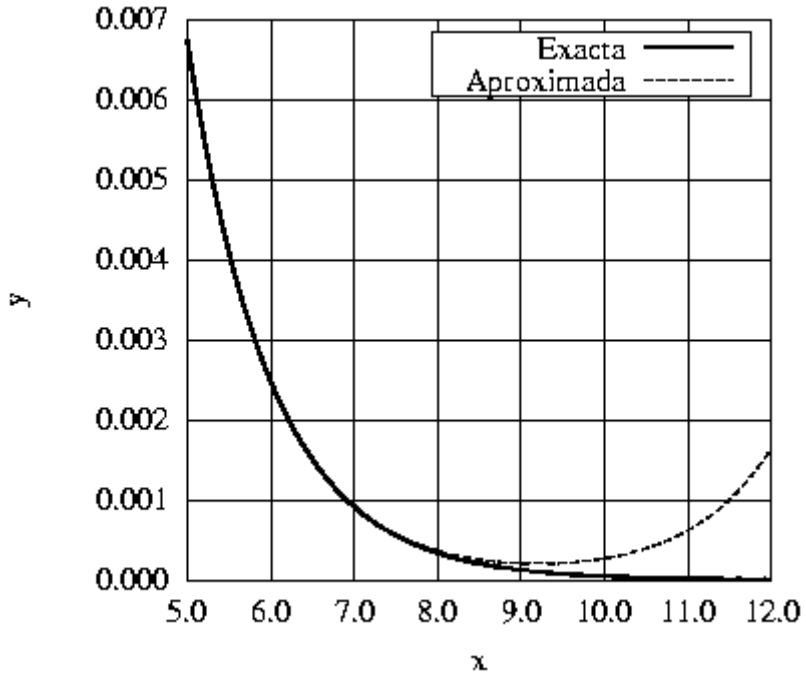

Figure: Representación gráfica de las funciones $y = 1.0 \cdot 10^{-8} e^{x} + e^{-x}$ e e^{-x} en donde se pone de manifiesto que ambas funciones difieren rápidamente a partir de un cierto valor de la ordenada x.

[scale=0.7]eps/sinu

El problema anterior se dice que es inherentemente inestable, o empleando una terminología más común en cálculo numérico, se dice que está 'mal condicionado' (*ill-conditioned*).