Capítulo 2

Números de Máquina y Número de Condición

1. Cuestiones previas

En el *álgebra lineal computacional* (o en el análisis numérico en general) interesan no solo los algoritmos sino su *confiabilidad* y su *costo computacional*. La primera cuestión aparece porque en la práctica no es posible en general trabajar con precisión infinita (lo que hace imprescindible el estudio de los errores numéricos y su propagación). La segunda cuestión tiene que ver con el número de operaciones involucradas en el algoritmo (*complejidad del algoritmo*) cuestión que impacta en el tiempo de ejecución[1].

Comenzamos entonces definiendo de manera informal los números de máquina.

Los números de máquina, son los que la máquina puede representar de forma exacta.

Notemos que -por su propia definición- podemos sospechar que los números de máquina son finitos y en consecuencia la *mayoría* de los números reales no podrán almacenarse sin cometer errores. Vamos a tratar de precisar este punto en lo que sigue. Vamos a asumir que trabajaremos con número reales o complejos, sin embargo como para almacenar un número complejo basta almacenar su parte real e imaginaria podemos enfocarnos en el caso real.

Consideremos un número cualquiera, por ejemplo $x = 125.4$. Estamos habituados a trabajar en base 10 lo que significa que estamos pensando a x en la forma

$$x = 1\,10^2 + 2\,10^1 + 5\,10^0 + 4\,10^{-1}.$$

Para número real (positivo) diferente, la escritura

$$\tilde{b}_k\tilde{b}_{k-1}...\tilde{b}_0.\tilde{b}_{-1}\tilde{b}_{-2}...$$

con $0 \leq \tilde{b}_i \leq 9$ y $\tilde{b}_k \neq 0$, la asociamos con

$$x = \sum_{-\infty}^{k} \tilde{b}_i 10^i.$$

La escritura hecha de este modo es única, salvo desarrollos periódicos del número 9 como ocurre en el caso

$$0.999... = 1,$$

en el que el mismo número, en este caso $x = 1$, puede escribirse de dos modos distintos. En este contexto, el número 10 se denomina la base y los $\tilde{b}_i$ los dígitos del desarrollo.

[1] Otras consideraciones son posibles: por ejemplo si puede o no paralelizarse.

2. Bases generales*

En general todo número $x \in \mathbb{R}$ $x \neq 0$, puede representarse en una base $b \in \mathbb{N}$, $b \geq 2$, de la forma[2]

$$(x)_b = sg(x)\tilde{b}_k\tilde{b}_{k-1}...\tilde{b}_0.\tilde{b}_{-1}\tilde{b}_{-2}... \tag{2.1}$$

donde los "dígitos" verifican $0 \leq \tilde{b}_i \leq b-1$, $sg(x)$ el signo de x y donde por convención numeramos los índices para que el punto -o coma- se ubique entre $\tilde{b}_0$ y $\tilde{b}_{-1}$.

Asumamos que x es positivo para no lidiar con el signo. La representación (2.1) se obtiene de la escritura de x como

$$x = \tilde{b}_k b^k + \tilde{b}_{k-1}b^{k-1} + ... + \tilde{b}_0 b^0 + \tilde{b}_{-1}b^{-1} + \tilde{b}_{-2}b^{-2} + ...$$

que es única si descartamos los desarrollos infinitos de la forma $\tilde{b}_j = b-1$ para todo $j \leq l$ con $l \in \mathbb{Z}$ fijo.

Para aclarar este punto recordemos la expresión cerrada de la geométrica

$$\sum_{j=0}^{m} \beta^j = \frac{\beta^{m+1}-1}{\beta-1},$$

válida para $\beta \neq 1$. Notemos que en caso de tener un x con un desarrollo de la forma

$$x = \sum_{i=-\infty}^{l} (b-1)b^i,$$

podemos reescribirlo como

$$x = (b-1)b^l\left(\sum_{i=-\infty}^{0} b^i\right) = (b-1)b^l\frac{b}{b-1} = b^{l+1},$$

lo que indica dos representaciones alternativas para el mismo x. Como ejemplo si $l=0$ tenemos

$$10 = (x)_b = (b-1).(b-1)(b-1)(b-1)\cdots$$

Continuemos asumiendo que $x > 0$. El uso del punto *fijo* en (2.1) determina la división entre su *parte entera*[3] $[x] \in \mathbb{Z}$ y la diferencia $0 \leq x - [x] < 1$. En efecto, por un lado, para todo $m \in \mathbb{N}$

$$\sum_{0}^{m} \tilde{b}_i b^i \in \mathbb{Z}$$

y

$$0 \leq \sum_{-\infty}^{-1} \tilde{b}_i b^i < \sum_{-\infty}^{-1} \tilde{b}_i(b-1) = (b-1)\frac{1}{b-1} = 1.$$

Por otro lado, esta representación tradicional no da una idea rápida del orden de magnitud de un número. Por esta razón se suele trabajar con una versión *normalizada* eligiendo ubicar el punto justo a la izquierda de $\tilde{b}_k$ (i.e. el primer dígito no nulo de x). Así, un $0 \neq x \in \mathbb{R}$, puede escribirse (renombramos los dígitos eliminando el tilde y los índices)

[2] $(x)_b$ se lee x en la base b.

[3] La parte entera de x, denotada $[x]$ se define como el mayor entero menor o igual a x.

$$(x)_b = sg(x)\, 0.b_0b_1b_2...\, b^e \tag{2.2}$$

donde, como antes, $0 \leq b_i \leq b-1$ y $e \in \mathbb{Z}$. Una vez más, esta representación es única si asumimos que $b_0 \neq 0$, y que en el caso $b_i = b-1$ para $i \leq l$, $b_{i+1} < b-1$ convenimos en tomar

$$(x)_b = sg(x)\, 0.b_0b_1...(b_{i-1}+1)\, b^{e+1}.$$

Suele llamarse notación científica normalizada a la escritura (2.2). La expresión $0.b_0b_1b_2...$ se llama *mantisa* y e *exponente*. Con esta normalización se ve a simple vista que $b^e < x \leq b^{e+1}$.

Observemos que el exponente es un entero que admite asimismo una representación en la misma base b

$$e = sg(e)\, c_lc_{l-1}...c_0$$

donde una vez mas $c_l \neq 0$ y $e = sg(e)\, \sum_0^l c_ib^i$.

3. Números de Máquina

En una máquina, la memoria dedicada para el almacenamiento de los números es limitada dedicando cierta cantidad de dígitos, digamos $m \geq 1$, para la mantisa y cierta cantidad, digamos $E \geq 1$, para el exponente. Los números que pueden representarse de forma exacta con esas limitaciones se denominan *números de máquina*.

La cantidad de números de máquina es obviamente finita. De hecho podemos calcular fácilmente su cantidad: considerando que $b_0 \neq 0$ tenemos $b-1$ posibilidades para la elección del b_0, y b posibilidades para cada uno de los restantes b_i, $1 \leq i \leq m-1$. La cantidad diferente de mantisas es entonces $(b-1)b^{m-1}$ (para el número cero se utiliza un símbolo especial). Análogamente hay b^E distintos exponentes. Considerando los respectivos signos de la mantisa y el exponente, habrá $4(b-1)b^{m+E-1}$ diferentes números de máquina.

El mayor exponente para una máquina dada es el número

$$E_{max} = \sum_{j=0}^{E-1}(b-1)b^j = \frac{b^E-1}{b-1}(b-1) = b^E - 1$$

y la mayor mantisa

$$M_{max} = \sum_{j=1}^{m}(b-1)b^{-j} = \frac{(b-1)}{b}\sum_{j=0}^{m-1} b^{-j} = \frac{(b-1)}{b}\frac{(b^{-m}-1)b}{1-b} = 1 - b^{-m},$$

lo que da

$$M_{max}^{E_{max}} = \left(1 - b^{-m}\right) b^{b^E-1},$$

como mayor número de máquina (análogamente se obtiene el menor como $-M_{max}^{E_{max}}$).

Cualquier número mayor a $M_{max}^{E_{max}}$ producirá un desborde *overflow*. El *menor* número *positivo* de máquina es obviamente $b^{-E_{max}-1}$. Todo número x, $0 < x < b^{-E_{max}-1}$ genera un desborde por debajo *underflow*.

FIGURA 1. Algunos números positivos de máquina con $b = 2$ y $m = 4$. Notar que los números son equiespaciados únicamente entre potencias sucesivas de 2, y que la cantidad de números de máquina en esos rangos se mantiene constante.

Escalas de los números de máquina: En las máquinas que utilizamos habitualmente $b = 2$ y $m = 52$ y $E = 10$. Esto es, se trabaja en base 2 y se utilizan 64 bits para almacenar un número (esta formato se denomina *doble precisión*[a]). El término *bit* significa dígito binario (admite solo dos caracteres: 0 y 1). De los 64 bits disponibles, dos se reservan para los signos de la mantisa y el exponente.
En este caso, considerando que $2^{10} = 1024 \sim 10^3$:

$$M_{max}^{E_{max}} = (1 - 2^{-52})2^{2^{10}-1} \sim 2^{1000} \sim (2^{10})^{100} \sim 10^{300}.$$

Esta máquina puede trabajar con números del orden de 10^{300} sin overflow. En la escala pequeña puede trabajar (haciendo una cuenta similar para $b^{-E_{max}-1}$) con números del orden 10^{-300}. Estas cantidades son razonables para describir las mayoría de las magnitudes que aparecen en las disciplinas científicas. Notemos que la mayor y menor escala se deciden básicamente en términos del tamaño del exponente y *no del tamaño de la mantisa*. Aumentar los bits de la mantisa no mejora las escalas (i.e. no aumenta el rango de valores extremales de nuestra máquina).

[a]En *precisión simple* se usan 32 bits, con $E = 7$ y $m = 23$.

Si un número $x \in \mathbb{R}$ no es de máquina, se puede elegir el número de máquina mas cercano para representarlo (*redondeo*) o, por ejemplo, el inmediato inferior (*truncado*). Dado un número real x denominamos $fl(x)$ al número de máquina elegido para representarlo. Se utiliza fl para indicar que estamos trabajando en *punto flotante*. El hecho de "mover" el punto de acuerdo a (2.2) acarrea una consecuencia muy importante a la hora de truncar o redondear el número y es la de mantener uniforme el *error relativo* a la hora de almacenar x a lo largo de todas la escalas. Esta idea es central y no podría lograrse eligiendo números de máquina equiespaciados.

Errores Relativos y Absolutos En una magnitud escalar x_v, el error absoluto se define como

$$e_{abs} = |x_a - x_v|,$$

donde x_a es el valor aproximado de x_v. En general (y para $x_v \neq 0$) estaremos interesados en el error relativo

$$e_{rel} = \frac{|x_a - x_v|}{|x_v|},$$

es decir en el tamaño del error absoluto respecto del tamaño del valor exacto x_v. Para magnitudes vectoriales o matriciales reemplazamos el módulo por una norma.

Distribución de los números de máquina: Los números de máquina no se distribuyen de manera uniforme. Describamos dos números de máquina consecutivos $x_1 < x_2$ asumiendo, para simplificar, que $0 < x_1$ y que se escribe:

$$(x_1)_b = 0.b_0b_1b_2...b_{m-1}\, b^e.$$

Un momento de reflexión nos dice que el próximo número de máquina (salvo que haya overflow) será

$$(x_2)_b = 0.b_0b_1b_2...(b_{m-1}+1)\, b^e$$

salvo que $b_{m-1} = b-1$ en cuyo caso

$$(x_2)_b = 0.b_0b_1b_2...(b_{m-2}+1)0\, b^e$$

salvo que también $b_{m-2} = b-1$ en cuyo caso habrá que repetir el argumento y eventualmente -en caso de que $b_i = b-1$ para todo $0 \leq i \leq m-1$

$$(x_2)_b = 0.1b^{e+1}.$$

En cualquier caso observamos que

$$x_2 - x_1 = b^{-m}b^e$$

no es constante al variar el exponente e, pero sí lo es para cada exponente fijo dado.

La consecuencia mas notable de la distribución de los números de máquina es que si $x \in \mathbb{R}$, y existen $x_1 < x_2$ números de máquina tales que $x_1 \leq x \leq x_2$ (i.e. x esta dentro de las escalas manejadas por la máquina) entonces

$$|x - fl_{redond}(x)| \leq \frac{1}{2}b^{e-m}$$

$$|x - fl_{trunc}(x)| \leq b^{e-m}$$

para los casos de redondeo y truncado respectivamente.

Error relativo y precisión de la máquina: Nos concentraremos en el caso de redondeo, por lo que asumiremos de aquí en mas que $fl = fl_{redond}$. Queremos acotar el error relativo al almacenar un número $x \neq 0$. Observando que

$$b^{e-1} = |0.1\, b^e| \leq |x|,$$

resulta

$$|\frac{x - fl(x)}{x}| \leq \frac{1}{2}b^{1-m}.$$

Notar que el error relativo depende *del tamaño de la mantisa*. En el caso de base $b = 2$ y $m = 52$ mencionado antes resulta:

$$|\frac{x - fl(x)}{x}| \leq 2^{-52} \sim 10^{-16}.$$

este número, propio de cada máquina, se denomina: precisión de la máquina o épsilon de la máquina y se denota con ε.
En el caso del ejemplo, en terminos simples, dice que nuestra máquina almacena 16 dígitos exactos (en base 10). Naturalemente sería ideal conservar esta precisión a lo largo de las operaciones que debamos realizar a partir de $fl(x)$. Eso, como veremos a continuación, no es posible en general.

Como acabamos de ver, en general ocurre que $x \neq fl(x)$[4]. . Sin embargo podemos garantizar que la diferencia verifica

$$x - fl(x) = \mu_x x,$$

con

$$\frac{|x - fl(x)|}{|x|} = |\mu_x| \leq \varepsilon.$$

Veamos entonces que ocurre con estos errores al efectuar alguna operación sencilla como por ejemplo sumar. Vamos a distinguir la operación realizada por la máquina con el símbolo $\oplus$.

Pretendemos calcular $x+y$, de modo simplificado la máquina realiza las siguientes operaciones: primero

$$x \to fl(x) \qquad y \to fl(y)$$

luego suma $fl(x) + fl(y)$ y finalmente almacena el resultado

$$fl(x) + fl(y) \to fl(fl(x) + fl(y)).$$

Cómo se comportará el error resultante?. Por una lado tenemos:

$$fl(x) = x(1 + \mu_x), \qquad fl(y) = y(1 + \mu_y y),$$

por lo que

$$fl(fl(x) + fl(y)) = (fl(x) + fl(y))(1 + \mu_z)$$

y en definitiva

$$x \oplus y = fl(fl(x) + fl(y)) = (x(1 + \mu_x) + y(1 + \mu_y y))(1 + \mu_z).$$

Vemos entonces que *si* $0 < x, y$ es posible acotar

$$|x \oplus y - (x + y)| \leq (x + y)2\mu + O(\varepsilon^2)$$

con $|\mu| \leq \varepsilon$. Es decir que hemos de algún modo preservado el tamaño del error relativo[5] Como es bien sabido, eso no es posible si *restamos* números similares[6]. Un ejemplo elemental es el siguiente: tomemos $b = 10$, $m = 4$, la precisión es $\varepsilon = \frac{1}{2}10^{-3} = 0.0005$. Si $x = 125,49$ e $y = 125,31$ tenemos respectivamente $x - y = 0.18$ $x \ominus y = 0.2$ por lo que el error relativo es

$$|\frac{x - y - x \ominus y}{x - y}| = \frac{0.02}{0.18} \sim 0.11,$$

es decir que a pesar de que $\varepsilon \sim 10^{-3}$ el error en la cuenta anterior es del 11 %. El ejemplo anterior muestra que en una simple operación podemos perder dígitos significativos (de hecho nuestra máquina no ha acertado ningún dígito de la solución). Como regla general deben evitarse restas de números similares[7] para evitar la denominada *cancelación catastrófica*.

Es fácil construir ejemplos de números de máquina $0 < x < y$ en los cuales no solo $x+y \neq x \oplus y$ sino que, por ejemplo, $x \oplus y = y$. En particular es fácil ver que

$$1 \oplus \varepsilon > 1, \tag{2.3}$$

y

$$1 \oplus \varepsilon/2 = 1$$

[4]Notemos que incluso números muy sencillos como $\frac{1}{10}$ no son de máquina y la introducción de un error de redondeo es inevitable. Evalúe en Python la expresión $0.1 + 0.1 + 0.1! = 0.3$, qué obtiene?.

[5]Cuando los errores se acumulan de forma aditiva, como en este caso, estamos en un buen escenario porque harían falta una enorme cantidad de operaciones para deteriorar el error inicial.

[6]Verifique que la cuenta anterior no puede repetirse si los signos de x e y difieren.

[7]Ver la guía de ejercicios para mas ejemplos

lo que da una posible definición alternativa del ϵ como el menor número de máquina con la propiedad (2.3). Otras cuestión que aparece entonces en la aritmética de la máquina es la pérdida de la propiedad asociativa, ya que usando los comentarios previos vemos que por ejemplo

$$(1 \oplus \varepsilon/2) \oplus \varepsilon/2 = 1 \neq 1 \oplus (\varepsilon/2 + \varepsilon/2).$$

Hay diversas fuentes de errores computacionales que pueden estropear completamente el resultado de los algoritmos. En este sentido hay dos conceptos que nombraremos tangencialmente ya que no son objeto central de este curso: la *condición* y la *estabilidad*.

4. Condición y Estabilidad

De manera muy general, un *problema bien condicionado* es aquél que reacciona benignamente con los errores en los datos. Es decir, que sus soluciones no varían demasiado al no variar demasiado los datos. Por el contrario, si un problema está muy mal condicionado, no lograremos resolverlo con mucha precisión aunque limitemos los errores en los datos (salvo que trabajemos con precisión infinita). La noción de condición es algo *intrínseco del problema* y está mas allá de nuestro algoritmo de resolución. Por otra parte, cuando los problemas están bien condicionados tenemos esperanza de resolverlos con precisión siempre que nuestro algoritmo no incremente desproporcionadamente los errores inherentes a los datos. En este caso hablaramos de *algoritmos estables* y por el contrario, de *algoritmos inestables* si no cumplen con este criterio. La estabilidad entonces es algo *intrínseco del algoritmo*.

Es posible dar una expresión precisa para la noción de condición, a través del llamado *número de condición*. Consideremos el problema de evaluar una función en el valor $x_0 \neq 0$. Si por alguna razón (error de medición o redondeo) modificamos el dato a evaluar x_0 en un una cierta magnitud pequeña h entonces el error relativo que cometeremos es (asumiendo que la función es de continuamente diferenciable)

$$\frac{f(x_0+h) - f(x_0)}{f(x_0)} = \frac{hf'(\eta)}{f(x_0)},$$

para cierto η intermedio entre x_0 y $x_0 + h$. Esto indica que para $h \sim 0$

$$\frac{hf'(\eta)}{f(x_0)} \sim \frac{x_0 f'(x_0)}{f(x_0)} \frac{h}{x_0}$$

el error relativo en los datos $\frac{h}{x_0}$ se magnifica en nuestro problema en un factor

$$\frac{|x_0||f'(x_0)|}{|f(x_0)|},$$

llamado el *número de condición* de evaluar f en x_0. Siguiendo un razonamiento similar vemos que para la versión $f : \mathbb{R}^n \to \mathbb{R}^m$ puede definirse el número de condición de este problema de la forma[a]

[a]Un ejemplo concreto y elemental de mal condicionamiento es, como hemos visto, la resta de números similares: si escribimos $f : \mathbb{R}^2 \to \mathbb{R}$, $f(x,y) = x - y$, se tiene $Df(x_0, y_0) = (1, -1)$, $\|(1,-1)\|_\infty = 2$, $\|(x_0, y_0)\|_\infty = max\{|x_0|, |y_0|\}$, luego $\frac{\|x_0\|\|Df(x_0)\|}{\|f(x_0)\|} \sim \frac{1}{\|x_0 - y_0\|}$.