TEMA 1.

ERRORES DE REDONDEO Y ARITMÉTICA DE PRECISIÓN FINITA

1. Introducción

Las prestaciones de los algoritmos para resolver muchos problemas dependen en gran medida de la exactitud con la que se pueden representar los números reales en la máquina o soporte donde se han de implementar. Para diseñarlos y codificarlos en un lenguaje que entienda la máquina, es necesario conocer como opera la aritmética de esa máquina.

Los ordenadores no guardan la información relativa a un número con precisión infinita, sino mediante una información empaquetada en grupos de bits, denominados bytes. Casi todos permiten al usuario escoger entre diversas formas de representar un determinado número. Estas representaciones varían casi siempre en función del número de bits utilizados o de si se hace en formato entero, coma flotante (real) etc.

2. Nomenclatura

β	base de numeración de la máquina
F	sistema de numeración
fl(x)	representación del numero x en un ordenador
L	límite de underflow
t	precisión de la máquina
U	límite de overflow

3. Representación de un número en un ordenador

La representación de un número entero en un ordenador es exacta, siempre y cuando su valor está comprendido dentro de los límites admisibles por este. Por el contrario en formato real sólo es posible representar un conjunto de números F. Este conjunto F, denominado sistema de numeración, está caracterizado por cuatro parámetros:

- β la base de numeración de la máquina
- t la precisión de la máquina
- U el límite de overflow
- L el límite de underflow

El conjunto F está compuesto por todos los números reales f de la forma:

$$f = \pm\left(\frac{d_1}{\beta}+\frac{d_2}{\beta^2}+\ldots+\frac{d_t}{\beta^t}\right)\cdot \beta^e \qquad \text{(ER1)}$$

donde los números enteros $d_1 \ldots d_t$ satisfacen $0 \le d_i \le \beta - 1$, i=1,...,t.

También se ha de cumplir que $L \le e \le U$.

Al número entero 'e' se le denomina exponente y a $\left(d_1/\beta+\ldots+d_t/\beta^t\right)$ mantisa.

Obsérvese que se cumple $m \le |f| \le M$, donde

$$m = \beta^{L-1}; \qquad M = \beta^U\left(1-\beta^{-t}\right) \qquad \text{(ER2)}$$

Es importante señalar la matisa tiene exactamente t sumandos, ni uno más ni uno menos.

Considere, por ejemplo, una máquina hipotética que opere en base 10 (β=10) y con precisión 3 (t=3), donde queremos representar los números 0.301 y 3.01.
La representación de estos numeros será pues:

$$\left(\frac{3}{10}+\frac{0}{10^2}+\frac{1}{10^3}\right)10^0 = 0.301; \qquad \left(\frac{3}{10}+\frac{0}{10^2}+\frac{1}{10^3}\right)10^1 = 3.01$$

Ambos números son representables por la máquina considerada. Destacar que la mantisa es la misma en ambos casos y que sólo cambia el valor del exponente.

4. Truncamiento y redondeo

¿Cuál sería, sin embargo, la representación del número 3.246 en la máquina anterior?. Lo primero que debemos darnos cuenta es que dicho número no es representable por la máquina. Para que fuese representable necesitaríamos una máquina que tuviese una precisión de al menos t=4.

En este caso el ordenador lo que hace es representar un número cercano que sí sea representable, en este caso tenemos dos posibilidades, o bien usar *truncamiento* o bien usar *redondeo*.

Un número en un ordenador será representado por el número más cercano del conjunto F de tal manera que fl(x) es el elemento $c \in F$ más cercano de x si se

usa redondeo. Sin embargo, si se usa truncamiento fl(x) es el elemento $c \in F$ más cercano a x que satisfaga $|c| \leq |x|$.

En otras palabras, en nuestra máquina hipotética el numero 3.246 se representaría como 3.24 si se usa truncamiento o bien como 3.25 si se usa redondeo. El truncamiento simplemente elimina todas las cifras que no pueden ser representadas por la máquina. El resondeo, por otra parte, toma la primera cifra no representable (en nuesrto ejemplo el 6) y le suma $\beta/2$. Si el resultado es superior o igual a β entonces añade una unidad al último digito que sí era representable y al resultado le aplica truncamiento, y si el el resultado es menor que β entonces aplica truncamiento. (en el ejmplo anterior al 6 le sumamos cinco unidades (10/2) como el resultado es 11, superior a 10, sumamos una unidad al dígito anterior (el 4), y aplicamos truncamiento.

La diferencia puede ser importante como obviamente se puede suponer. El truncamiento o redondeo da lugar a lo que se conoce comunmente como error de redondeo. Afecta a todas las operaciones que se realizan en un ordenador.
Se puede comprobar facilmente que dado un número x el error relativo si se usa redondeo es :

$$\frac{|x - fl(x)|}{|x|} \leq \frac{1}{2}\beta^{1-t} \qquad \text{(ER3)}$$

y si se usa truncamiento ::

$$\frac{|x - fl(x)|}{|x|} \leq \beta^{1-t} \qquad \text{(ER4)}$$

a la cantidad β^{1-t} se la denomina **precisión de la máquina o epsilon**, ε, su valor lo define la cantidad más pequeña que añadida a 1 tiene representación en la máquina. En la Figura 1 se comprueba el valor de epsilon en una máquina utilizando Matlab®.

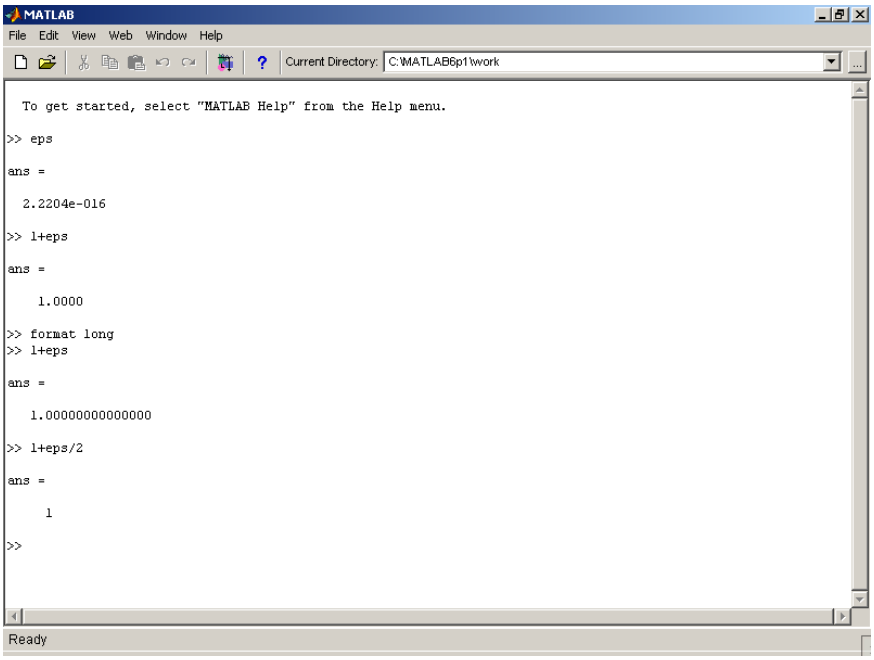

Figura 1. Estimación de Epsilon con Matlab®.

Para llevar a cabo operaciones -podemos pensar el el ordenador como una máquina que hace sumas, pero a muy alta velocidad- los números de un ordenador son en primer lugar convertidos a un formato de mantisa común, desplazando los digitos significativos de cada uno de ellos a posiciones equivalentes del otro.

Por ejemplo: supongamos que trabajamos en una máquina β=10, t=4. y que queremos realizar la siguiente suma: 416+1352. El primer paso es representar los números en coma flotante:

$$\left(\frac{4}{10}+\frac{1}{10^2}+\frac{6}{10^3}+\frac{0}{10^4}\right)10^3 + \left(\frac{1}{10}+\frac{3}{10^2}+\frac{5}{10^3}+\frac{2}{10^4}\right)10^4$$

Un ordenador no puede sumar los numeros tal cual están, de hecho el ordenador necesita hacer lo mismo que hacemos nosotros cuando sumamos numeros a mano. Al hacer una operación a mano (asumamos base 10 por comodidad) colocamos las unidades debajo de las unidades, las decenas debajo de las decenas etc, así en nuesrto ejemplo sumaríamos 6+2, comprobamos si el numero supera la base (10), si la supera o iguala arrastramos una unidad a la siguiente suma, y contimuamos con (1+5) etc... exactamente lo mismo hace el ordenador. Sin embargo, en la representación en coma flotante anterior, debido a que los exponentes son diferentes, no

tenemos alineadas unidades con unidades, decenas con decenas etc. Para poder hacer la suma debemos tener los números **en formato de *mantisa común***. O en otras palabras, transformar el número con el menor exponente a un formato donde el exponente de ambos sumandos sea el mismo. En nuestro ejemplo:

$$\left(\frac{0}{10}+\frac{4}{10^2}+\frac{1}{10^3}+\frac{6}{10^4}\right)10^4 \; + \\ \left(\frac{1}{10}+\frac{3}{10^2}+\frac{5}{10^3}+\frac{2}{10^4}\right)10^4$$

Con lo cual ya podemos realizar la suma.

Considere ahora la suma de 103.0 + 1.036 en una máquina t=4, β=10.

$$\left(\frac{1}{10}+\frac{0}{10^2}+\frac{3}{10^3}+\frac{0}{10^4}\right)10^3+\left(\frac{1}{10}+\frac{0}{10^2}+\frac{3}{10^3}+\frac{6}{10^4}\right)10^1$$

cambiando a formato de mantisa común:

$$\left(\frac{1}{10}+\frac{0}{10^2}+\frac{3}{10^3}+\frac{0}{10^4}\right)10^3+\left(\frac{0}{10}+\frac{0}{10^2}+\frac{1}{10^3}+\frac{0}{10^4}\right)10^3=104.0$$

Es fácil darse cuenta de que ha habido una pérdida de cifras significativas importante. De esta forma si ese desplazamiento supera los digitos significativos que la máquina permite se puede perder toda la información del número desplazado. Este hecho se conoce como ***cancelación catastrófica o simplemente error de cancelación*****,** y surge al operar con números de muy diferentes órdenes de magnitud (numeros muy grandes y muy pequeños) el concepto de número grande y número pequeño es relativo a la precidión de la máquina. Por ejemplo, MATLAB ($t \approx 16$) hace la operación anterior sin ningún problema. Sin embargo es totalmente imposile sumar en MATLAB números que difieran en 16 ordenes de mágnitud. Si, por ejemplo, en MATLAB sumamos números que difieren en 8 cifras significativas estamos perdiendo 8 cifras en el número más pequeño (ver Figura 2).

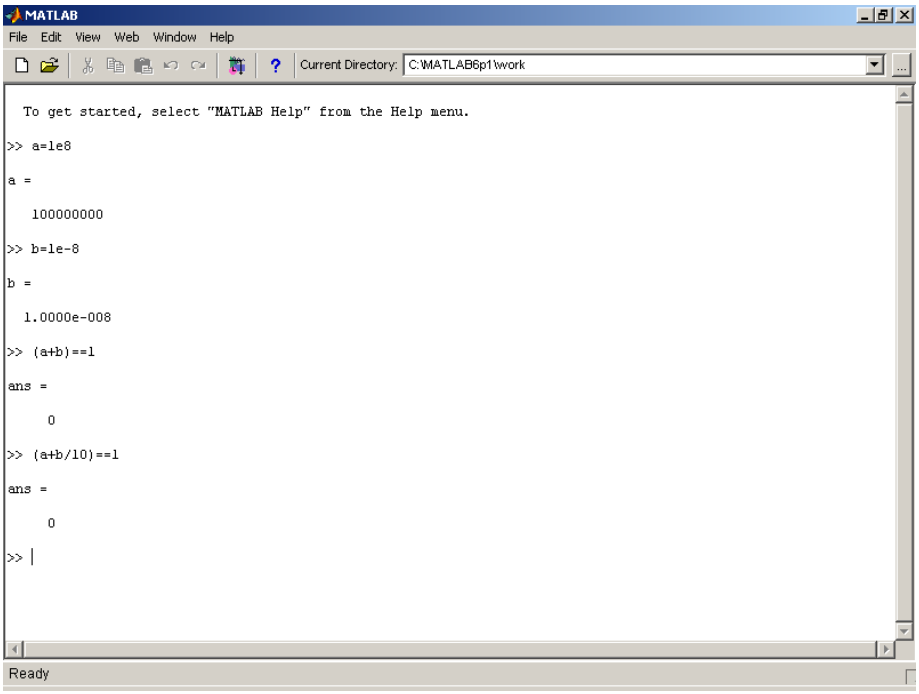

Figura 2. Error de cancelación en Matlab®.

Otro aspecto importante que los cálculos de la aritmética con precisión finita puede introducir lo constituye el cálculo de la diferencia de dos números casi iguales. Por ejemplo en un ordenador con $\beta=10$ y $t=3$ que use truncamiento la diferencia de 0.2500 y de 0.2499 se calcularía de la siguiente manera: $0.2500-0.2499=fl(fl(0.2500)-fl(0.2499))=fl(0.250-0.249)=0.001$. La diferencia real es 10 veces menor que la calculada.

5. Error de truncamiento y de redondeo

El **error de redondeo** es el que resulta de reemplazar un número por su forma de punto flotante, es decir, por su representación en una máquina concreta. Este error se denomina de redondeo tanto para aproximación por truncamiento como por redondeo.

El **error de truncamiento** ocurre cuando un proceso que requiere un número infinito de pasos se detiene en un número finito de pasos. Generalmente se refiere al error involucrado al usar sumas finitas o truncadas para aproximar la suma de una serie infinita. Notese que el error de truncamiento, a diferencia del error de redondeo, no depende directamente del sistema numérico que se emplee.

Por lo tanto habrá que tener mucho cuidado con las singularidades que puedan presentar los problemas a resolver en un determinado algoritmo, los criterios con los que se decida cuando se ha de parar un determinado proceso iterativo, la precisión máxima obtenible, cuando se ha de considerar que un número es cero, etc.

A modo de ejemplo imagine que se desea calcular cuanto vale $e^{-5.5}$. Esto se puede hacer considerando el desarrollo en serie de e^x y añadiendo sumandos hasta que el resultado, en nuestra máquina, no varíe.

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots \qquad \text{(ER5)}$$

Suponga que utilizamos una máquina β=10, t=5, más de lo que utilizamos habitualmente cuando hacemos cálculos a mano.

$e^{-5.5}$ = +1.0000 – 5.5000 + 15.125 –27.730 + 38.129 – 41.942 + 38.446 – 30.208 + 20.768 – 12.692 + 6.9803 – 3.4902 + 1.5997 - … = +0.0026363

La suma termina después de 25 términos pues los siguientes no aportan dígitos significativos al resultado. El valor real de la operación es, sin embargo, 0.00408677. Como se puede ver la diferencia es muy importante, tanto que ambos resultados apenas se parecen.

6. Cálculo de derivadas por diferencias finitas

Uno de los aspectos más importantes de los razonamientos anteriores, en cuanto a cálculo numérico se refiere, es el relativo al cálculo de derivadas por diferencias finitas.

Previendo estas dificultades es conveniente modificar el método de tal manera que se soslaye esta dificultad. La primera modificación que se puede hacer surge de la aplicación inmediata de la definición de derivada de una función en un punto (ver Figura 3):

$$f'(x_k) = \lim_{\Delta x \to 0} \frac{f(x_k + \Delta x) - f(x_k)}{\Delta x}$$

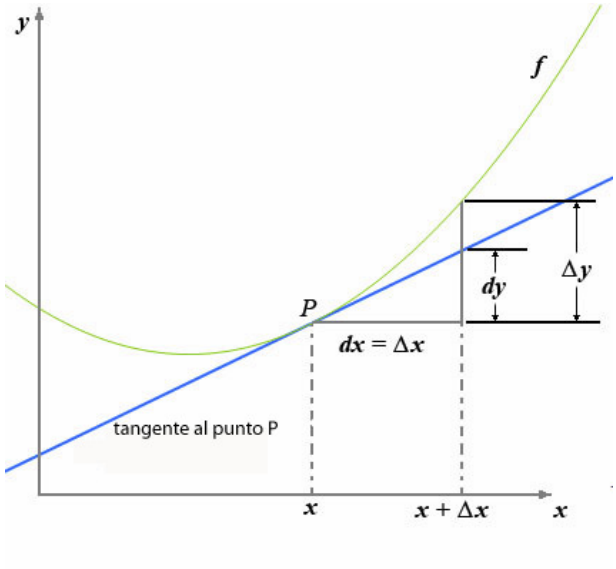

Figura 3. Cálculo de una derivada en un punto P.

en lugar de tener que saber la derivada de la función, se calcula su valor en un punto mediante la fórmula anterior utilizando un parámetro Δx adecuado. A la variante del método que surge de esta idea se la conoce como método de Newton por diferencias finitas.

Aunque no se expondrá la demostración, la convergencia por el método de diferencias finitas es buena. Sin embargo el problema se plantea al escoger el valor de Δx. Aunque en teoría la mejor aproximación a la derivada se obtendría cuando Δx fuese lo más pequeño posible, debemos tener en cuenta las consideraciones expuestas anteriormente sobre los errores de redondeo y los cálculos de precisión finita. De tal manera que el parámetro Δx debe escogerse lo suficientemente grande para evitar que:

$$fl(x+\Delta x)=fl(x)$$

o incluso si $fl(x+\Delta x)\neq fl(x)$, dado que la función 'f' es continua y su derivada también, al evaluar la función en dos puntos muy próximos, hay que evitar que: $fl(f(x+\Delta x))\neq fl(f(x))$ y aún así si Δx es pequeño si $f(x+\Delta x)$ es muy parecido a f(x) se perderán muchos de los dígitos significativos.

Por otra parte el valor de Δx no debe ser excesivamente alto porque estamos intentando sustituir el valor por la derivada.

Una regla sencilla que alcanza el compromiso entre ambos extremos es escoger Δx de tal manera que cumpla la regla empírica:

$$|\Delta x| = \sqrt{\varepsilon} \bullet \max\{tipx, |x_k|\}$$

donde "tip x" indica la magnitud típica de x.

Si el problema está bien escalado se puede usar $\Delta x = \sqrt{\varepsilon}$. No obstante cuando aparece mas de una variable con diversos órdenes de magnitud el uso de un mismo valor de Δx para todas ellas puede ser catastrófico y es por lo tanto mucho mejor calcular las derivadas utilizando un valor en cada caso.

7. RESUMEN

La representación de un número entero en un ordenador es exacta, siempre y cuando su valor está comprendido dentro de los límites admisibles por este. Esa representación está dada por una mantisa y un exponente, que tiene en cuenta la base de la máquina y su precisión.

Un número con más dígitos de la precisión de una máquina sera representado bien por truncamiento o por redondeo.

La precisión de la máquina o epsilon, ε, lo define la cantidad más pequeña que añadida a 1 tiene representación en la máquina.

El error de cancelación surge al operar con números de muy diferentes órdenes de magnitud (numeros muy grandes y muy pequeños).

El error de redondeo es el que resulta de reemplazar un número por su forma de punto flotante.

El error de truncamiento ocurre cuando un proceso que requiere un número infinito de pasos se detiene en un número finito de pasos.

Para el cáclulo numérico de una derivada es necesario encoger un incremento de x adecuado; una regla sencilla es escoger Δx de tal manera que cumpla la regla empírica:

$$|\Delta x| = \sqrt{\varepsilon} \bullet \max\{tipx, |x_k|\}$$

8. Derivadas complejas

Como se ha visto, la selección del valor de perturbación para el cálculo de una derivada no es un asunto trivial. Sin embargo, en software que admite variables complejas, como es el caso de MATLAB, que trabaja por defecto con algebra compleja es posible calcular numéricamente una derivada sin ningún tipo de error más allá de lo que la máquina sea capaz de representar. Por ejemplo, en MATLAB se puede conseguir una derivada numérica con aproximadamente 15 cifras significativas exactas.

Si en lugar de utilizar como perturbación para calcular la derivada el valor h se utiliza ih, donde i es la unidad imaginaria ($i = \sqrt{-1}$), desarrollando en serie de Taylor se obtiene que:

$$f(x^k + ih) = f(x^k) + ih\, f'(x^k) - \frac{h^2 f''(x^k)}{2!} - \frac{ih^3 f^{(3)}(x^k)}{3!} + \ldots$$

Y por lo tanto, despreciando los términos de orden 3 y superior se tiene que:

$$f'(x^k) \approx \frac{\mathrm{Im}\left[f(x^k + ih) \right]}{h}$$

A modo de ejemplo vamos a comparar, utilizando Matlab, el valor de la derivada que se obtendría, para diferentes valores de h (entre 0.01 y 10^{-19}) para la función $f(x) = x^{9/2}$ en el punto x = 1.5;

El valor exacto es 18.60081273425976

$f'(x^k) \approx \frac{f(x^k + h) - f(x^k)}{h}$	$f'(x^k) \approx \frac{\mathrm{Im}\left[f(x^k + ih) \right]}{h}$
18.81903084042103	18.59960712803634
18.62252574154422	18.60080067817763
18.60298294963592	18.60081261369893
18.60102974502453	18.60081273305415
18.60083443361305	18.60081273424770
18.60081491500409	18.60081273425964
18.60081280113946	18.60081273425976
18.60081422222493	18.60081273425976
18.60080978133283	18.60081273425976
18.60085419025381	18.60081273425976
18.60289700061912	18.60081273425976
18.58957432432362	18.60081273425977
18.56292897173262	18.60081273425976
20.42810365310288	18.60081273425976
0	18.60081273425976
0	18.60081273425976
0	18.60081273425976
0	18.60081273425976

9. Programación en MATLAB •

Cálculo de la derivada por incrementos finitos:

```
function d=derivada(f,x0)
% 'f' representa la función de Matlab que contiene la función a derivar
% x0 representa el punto donde se calcula la derivada.
f1=feval(f,x0); % Calculamos la función en el punto X0
h=sqrt(eps)*x0;% Valor de incremento de x
f2=feval(f,x0+h); % Calculamos la función en el punto incrementado
d=(f2-f1)/h; % Incrementos finitos
```

Por ejemplo, para el cálculo de la derivada de la función:

$$f(x) = -15\exp\left(\frac{123}{x}\right)$$

programamos la función de Matlab que contiene la función a derivar:

```
function f=cinetica(t,x)
f=-15*exp(-123/x);
```

e introducimos en la linea de comandos, por ejemplo, la orden:

```
d=derivada(@cinetica,0)
```

para evaluar en x=0 e introducir el resultado en la variable 'd'.