Tema 1: Preliminares (Parte I)

Prof. Marlliny Monsalve

Ingeniería Matemática y Computacional

Pontificia Universidad Católica de Chile

Álgebra Lineal Numérica

March 13, 2017

Álgebra Lineal

Álgebra Lineal (AL)

- Rama de las Matemáticas
- Estudio de espacios vectoriales y las transformaciones lineales entre estos

Espacios Vectoriales

Ejemplos de mucho interés: $\mathbb{R}^n$, $\mathbb{R}^{m\times n}$

Transformaciones Lineales

Sean U y V dos espacios vectoriales. Y sea T una función definida de U en V ($T : U \to V$). Se dice que T es una transformación lineal si

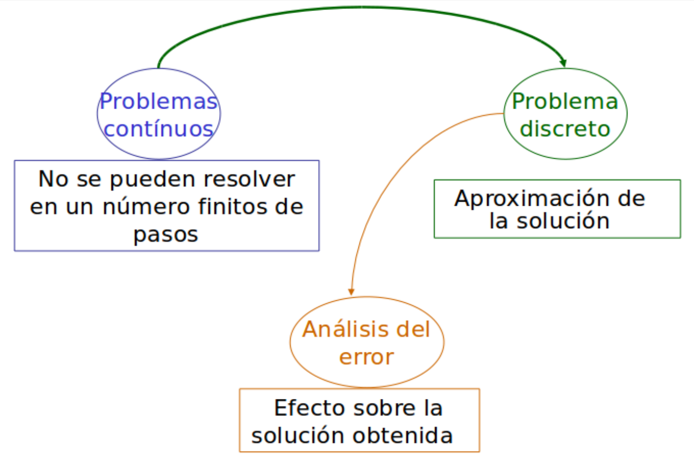
1. $T(0) = 0$
2. $T(\alpha x + y) = \alpha T(x) + T(y)$, $\forall x, y \in U$ y $\forall \alpha$ escalar

Álgebra Lineal Numérica (ALN) O Álgebra Lineal Aplicada

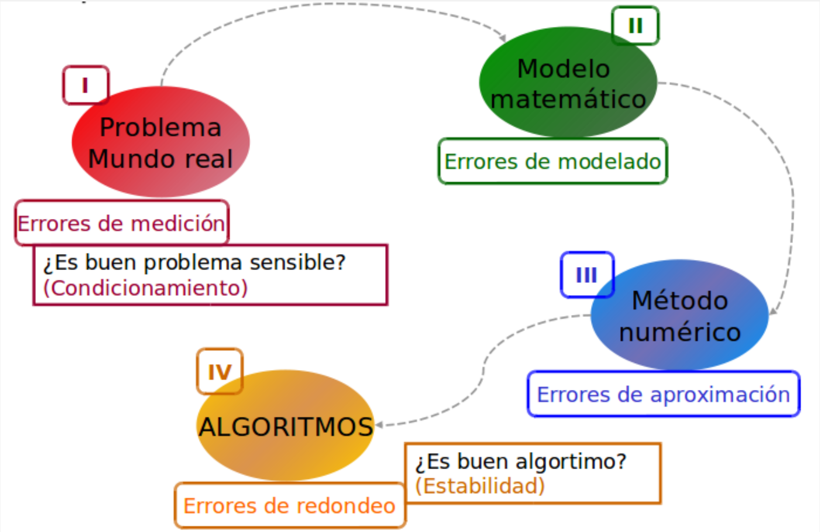
- Rama de las Matemáticas + Computación
- Visión simple pero útil: Estudio de Algoritmos para resolver problemas del AL!

Análisis / Álgebra NUMÉRICAS

Al usar Algoritmos puede haber errores!



Fuentes de error



Fuentes de error: Ejemplo

Problema/Solución

I: Predicción del clima

II: Sistema de ecuaciones diferenciales

III: Elementos finitos y Newton para sistemas no lineales

IV: Algoritmos

Errores

I: Errores de medición: Aparatos usados

II: Errores de modelado: Simplificaciones para hacer manejable el modelo.

III: Errores de aproximación: Solución$= \lim_{n\to\infty} G(n)$ donde G es un método iterativo y n es la cantidad de iteraciones.

IV: Errores de redondeo: Aritmética finita del computador

Fuentes de error: Ejemplo

Problema/Solución

I: Calcular el número e

II: Taylor: $e = \sum_{k=0}^{\infty} \frac{1}{k!}$

III: Truncar $e \approx \widehat{e} = \sum_{k=0}^{n} \frac{1}{k!}$

IV: Dado n; $\widehat{e} \leftarrow 0$

1: **for** $k = 0 : 1 : n$ **do**
2: $\quad \widehat{e} \leftarrow \widehat{e} + \frac{1}{k!}$
3: **end for**

Errores

I: Errores de medición:
NO!

II: Errores de modelado:
NO!

III: Errores de aproximación:
n no debe ser muy grande

IV: Errores de redondeo:
Con $k = 3$: $\frac{1}{6} = 0.1\overline{6}$

Fuentes de error: Ejemplo

Problema/Solución

I: Dada $A \in \mathbb{R}^{n\times n}$ y $b \in \mathbb{R}^n$. Hallar x tal que $Ax = b$

II: Si A es invertible, $x = A^{-1}b$

III: Muchos! Uno (tonto) Neumann: $A^{-1} = \sum_{k=0}^{\infty}(I - A)^k$

IV: Dado n; $T \leftarrow I$

1: **for** $k = 1 : 1 : n$ **do**
2: $\quad T \leftarrow T(I - A)$
3: **end for**
4: $\leftarrow T * b$

Errores

I: Errores de medición: NO!

II: Errores de modelado: NO!

III: Errores de aproximación: n debería ser muy grande: Costoso

IV: Errores de redondeo: Entradas de A y b.

Recordar

Contexto

- Dado el valor de f y algunas de sus derivadas en un punto c
- ¿Cómo hallar $f(x)$ con $x = c + \delta$? con δ dado.
- Note que x es un punto alrededor de c

Taylor: `taylor.m`

Si $f \in C^n[a, b]$ y si $f^{(n+1)}$ existe en (a, b), entonces para cada punto c y x en $[a, b]$ existe un punto ξ entre x y c tal que

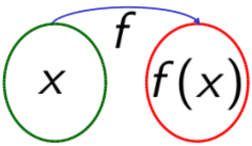
$$f(x) = f(c + \delta) = \sum_{k=0}^{n} \frac{1}{k!} f^{(k)}(c)(x - c)^k + E_n(x),$$

donde

$$E_n(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi)(x - c)^{n+1}.$$

Condicionamiento del problema

Problema: Hallar $f(x)$! (Abstracción)



x Entrada

f *Solver*

$f(x) = y$ Salida

¿Qué tan sensible es la Salida a pequeñas variaciones en la Entrada?

1. **Mal Condicionado:** Pequeños errores en la entrada generan grandes errores en la salida.
2. **Bien Condicionado:** Caso contrario.

$$\begin{aligned}
\tilde{x} &= x + \Delta_x \\
\tilde{y} &= f(\tilde{x}) = f(x+\Delta_x) = f(x) + f'(x)\Delta_x + O(\Delta_x^2) \approx y + f'(x)(\tilde{x} - x)
\end{aligned}$$

$$\underbrace{\left|\frac{\tilde{y} - y}{y}\right|}_{\substack{\text{Error}\\ \text{en salida}}} \lessapprox \underbrace{\left|x\frac{f'(x)}{f(x)}\right|}_{cond} \underbrace{\left|\frac{(\tilde{x} - x)}{x}\right|}_{\substack{\text{Error}\\ \text{en entrada}}}$$

Si *cond* es grande, puede que el problema esté **mal condicionado**

Condicionamiento del problema

¿Qué tan sensible es la Salida a pequeñas variaciones en la Entrada?

$$\frac{\text{Error Rel. en la salida}}{\text{Error Rel. en la entrada}} = \frac{|(f(\tilde{x}) - f(x))/f(x)|}{|(\tilde{x} - x)/x|} \tag{1}$$

Si (1) es:

1. ≈ 1, Error en salida es muy similar al de entrada $\therefore$ **Bien condicionado**
2. $>> 1$, Error en salida es mucho mayor al de entrada $\therefore$ **Mal condicionado**

$$\lim_{\Delta_x \to 0} \frac{|(f(\tilde{x}) - f(x))/f(x)|}{|(\tilde{x} - x)/x|} = \left| x \frac{f'(x)}{f(x)} \right| = cond$$

Y nuevamente concluimos que: Si *cond* es grande, puede que el problema esté **mal condicionado**

El número de condición es una propiedad del problema y NO depende del algoritmo usado para calcular la solución del mismo

Errores

$$
\begin{tikzcd}
x \arrow[r, "f"] \arrow[d] & f(x) \arrow[d] \\
\widetilde{x} \arrow[r, "f"] \arrow[dr, "\widetilde{f}"'] & f(\widetilde{x}) \arrow[d] \\
 & \widetilde{f}(\widetilde{x})
\end{tikzcd}
$$

- Problema: Hallar $f(x)$
- x: Data de entrada
- $f(x)$: Solución al problema con la data x
- $\widetilde{f}$: Método para aproximar f
- $\widetilde{f}(\widetilde{x})$: Salida del método $\widetilde{f}$ con la entrada $\widetilde{x}$

Errores

- ET: Error Total
- EP: Error de Propagación
- EC: Error de Cómputo

$$
\begin{aligned}
ET &= \widetilde{f}(\widetilde{x}) - f(x) \\
&= \widetilde{f}(\widetilde{x}) - f(x) + f(\widetilde{x}) - f(\widetilde{x}) \\
&= \underbrace{[f(\widetilde{x}) - f(x)]}_{EP} + \underbrace{[\widetilde{f}(\widetilde{x}) - f(\widetilde{x})]}_{EC}
\end{aligned}
$$

Errores

Comentarios

1. En *EP*: Diferencia entre las soluciones exactas pero con data de entrada distintas, $f(x)$ y $f(\widetilde{x})$, $\therefore$ **Error de Perturbación.**
2. En *EC*: Con la misma data de entrada, $\widetilde{x}$, el *EC* es la diferencia de la solución exacta $f(\widetilde{x})$ menos la solución aproximada $\widetilde{f}(\widetilde{x})$ $\therefore$ **Error de Computo.**
3. El método elegido no afecta al *EP*! sólo depende del condicionamiento del problema
4. El *EC* puede verse como la suma de los errores de aproximación y los errores de redondeo (Lámina Fuentes de error)
5. El análisis del error en la salida no es trivial!

Atención (Higham)

- ***Accuracy:*** *relative error of an approximate quantity* (Exactitud)
- ***Precision:*** *Precision is the accuracy with which the basic arithmetic operations are performed* (Precisión)

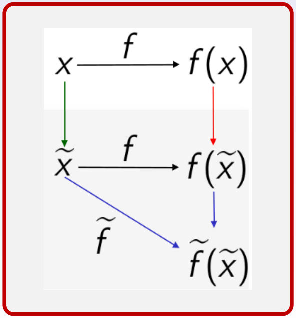
Errores

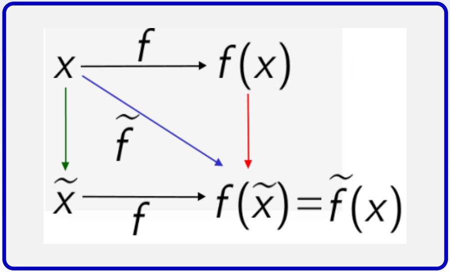
¿Cómo medir la calidad de la aproximación?

1. Queremos hallar $y = f(x)$ PERO:

   you can't always get what you want...

2. Supongamos que el método usará x para hallar la aproximación, i.e., $\widetilde{y} = \widetilde{f}(x)$
3. Ahora resulta útil preguntar ¿Para cuál data de entrada $\widetilde{x}$ la aproximación coincide con una solución exacta?, i.e, para qué $\widetilde{x}$ se cumple que $\widetilde{y} = f(\widetilde{x})$





Errores

Notación/Definición:

1. **Notación:** $\widetilde{x} = x + \Delta_x$ y $\widetilde{y} = y + \Delta_y \;\therefore$

$$y + \Delta_y = \widetilde{y} = \widetilde{f}(x) = f(\widetilde{x}) = f(x + \Delta_x)$$

2. **Definición:** Δ_x es el *Backward error* y Δ_y es el *Forward error* (Absolutos)

¿Cómo medir la calidad de la aproximación? (continuación...)

1. Si Δ_y es pequeño, la aproximación $\widetilde{y} = y + \Delta_y$ es muy cercana a la solución exacta del problema original $y = f(x)$
2. Si Δ_x es pequeño, la aproximación $\widetilde{y}$ es la solución exacta a un problema muy cercano al original, es decir, $\widetilde{y} = f(x + \Delta_x)$

***but** if you try sometimes well you just might find **You get what you need**:*

Errores: ejemplo didáctico no práctico!

Calcular e

$$f(x) = e^x = \sum_{k=1}^{\infty} \frac{x^k}{k!}$$

$$\widetilde{f}(x) = 1 + x + \frac{x^2}{2} + \frac{x^3}{3!}$$

USADOS CON $x = 1$

Backward Error

¿Cómo elegir $\widetilde{x}$ tal que $\widetilde{f}(x) = f(\widetilde{x}) = e^{\widetilde{x}}$?

$$\text{Si } \widetilde{f}(x) = e^{\widetilde{x}}, \text{ entonces } \ln \widetilde{f}(x) = \widetilde{x}$$
$$\ln \widetilde{f}(x) = \ln \widetilde{f}(1) = 0.980829 = \widetilde{x}$$

$$\begin{aligned} \Delta_x &= |\widetilde{x} - x| \\ &= |0.980829 - 1| \\ &= 0.019171 \end{aligned}$$

No se calculó e^1 (problema original) sino $e^{0.980829}$ (problema cercano al original)

Forward Error

$$\begin{aligned} \Delta_y &= |\widetilde{y} - y| \\ &= |\widetilde{f}(1) - f(1)| \\ &= |2.666667 - 2.718282| \\ &= 0.051615 \end{aligned}$$

Estabilidad: Backward y Forward

Definiciones

Un método para calcular $y = f(x)$ es...

1. *Backward estable* si $\forall x$, produce $\tilde{y}$ con un BE pequeño, es decir,

$$\tilde{y} = f(x + \Delta_x) \text{ con } \Delta_x \leq \epsilon$$

2. *Forward estable* si $\forall x$, produce $\tilde{y}$ con un FE pequeño, es decir,

$$\tilde{y} = y + \Delta_y \text{ con } \Delta_y \leq \epsilon$$

Verdades

- Para un problema bien condicionado: **Si un método es *Backward estable* entonces es *Forward estable*** (no viceversa)
- Encontrar cotas para el BE es lo que se conoce como *Backward error analysis* (BEA)
- En el BEA los errores de redondeo se interpretan a priori como perturbaciones de la data de entrada del problema.

Ejemplo para mantener en mente...

SEL: La no singularidad de la matriz es clave

- Entrada: A, b
- Problema: Hallar x_* tal que $Ax_* = b$
- Sol. exacta: x_*
- Aproximación: $\widetilde{x} = x_* + \Delta_{x_*}$

Sea $r = b - A\widetilde{x} = b - A(x_* + \Delta_{x_*}) = b - Ax_* - A\Delta_{x_*} \Rightarrow \boxed{r = -A\Delta_{x_*}}$

Por tanto, si A es no singular y $r = 0$ entonces $\Delta_{x_*} = 0$: BE igual a cero!

Equivalentemente $A\widetilde{x} = A(x_* + \Delta_{x_*}) = Ax_* + A\Delta_{x_*} \Rightarrow \boxed{A\widetilde{x} = b - r}$

Por tanto, si $r \approx 0$ entonces $\widetilde{x}$ es la solución a un SEL muy cercano al original!

Condicionamiento/Estabilidad

Condicionamiento $\rightsquigarrow$ Problema

Dice que tan sensible es el problema a las perturbaciones en el data de entrada.

Estabilidad $\rightsquigarrow$ Algoritmo

Dice que tan sensible es la salida del algoritmo a perturbaciones en el data en el proceso de cómputo.

Comentarios

$$\boxed{\left|\frac{\tilde{y}-y}{y}\right| \lessapprox \underbrace{\left|x\frac{f'(x)}{f(x)}\right|}_{cond}\left|\frac{(\tilde{x}-x)}{x}\right|}$$

$\Rightarrow$ FE relativo $\lessapprox$ *cond* $\times$ BE relativo

- La estabilidad (*backward*) del algoritmo garantiza que la aproximación es exacta para un problema cercano al original PERO no garantiza que la aproximación sea cercana a la solución exacta, *a menos que el problema esté bien condicionado*
- La aproximación obtenida para un problema mal condicionado *puede tener* un FE grande.

Condicionamiento/Estabilidad

Comentarios

- *Accuracy depends on the conditioning of the problem as well as the stability of the algorithm*
- Algoritmo Estable + Problema mal condicionado = (podría) Aprox. *Inaccuracy*
- Algoritmo Inestable + Problema bien condicionado = (podría) Aprox. *Inaccuracy*
- Tanto el Condicionamiento como la Estailidad están relacionados a pertubaciones de la data de entrada: Dependen de la data de entrada

Condicionamiento $\rightsquigarrow$ Problema

Ejemplo calcular $f(x) = e^x$

$$cond = \left| \frac{xf'(x)}{f(x)} \right| = |x|$$

Bien condicionado para x pequeño

Mal condicionado para x grande

Estabilidad $\rightsquigarrow$ Algoritmo

1: Dados $(\alpha; \beta) \rightsquigarrow (\frac{13}{3}; \frac{4}{3}); (\frac{3}{4}; \frac{5}{36})$

2: $x_0 := 1;\ x_1 := \frac{1}{3}$

3: **for** $k = 1 : 1 : n$ **do**

4: $\quad x_{k+1} = \alpha x_k - \beta x_{k-1}$

5: **end for**

Estable para $\alpha \leq 1$. Inestable $\alpha > 1$

Aritmética del Computador (repaso veloz!)

Notación Científica Normalizada (NCN)

Todo $x \in \mathbb{R}$: se puede representar como

$$x = (0.d_1 d_2 \dots d_t d_{t+1} \dots)_\beta \times \beta^e,$$

donde

- β es la *base de representación*
- Los dígitos $d_1 d_2 \dots d_t d_{t+1} \dots$ son la *mantisa*
- e es el *exponente* y $e \in (-\infty, +\infty)$
- Si $d_1 \neq 0$, notación *normalizada*: Para garantizar UNICIDAD!

$$0.345 \times 10^3 = 0.0345 \times 10^4 = \dots$$

Sistema punto flotante

- e es el *exponente* y $e \in [L, U]$ con L, U dados
- *Mantisa* con sólo t dígitos
- $\beta = 2$ es la *base de representación*
- Normalizada $d_1 \neq 0$.

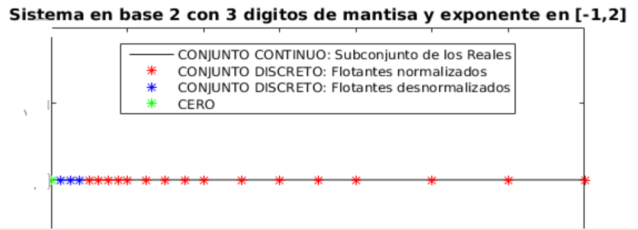
$$\mathbb{F}_n = \{x \in \mathbb{R} : x = \pm(0.d_1 d_2 \dots d_t)_\beta \times \beta^e\}$$

Sistema punto flotante

- $\mathbb{F}_n$ discreto finito: Posee $2(\beta-1)\beta^{t-1}(U-L+1)$ elementos normalizados
- Elemento más grande: $max = \beta^U(1-\beta^{-t})$ (umbral de *overflow*)
- Elemento más pequeño (normalizado): $min = \beta^{L-1}$ (umbral de *underflow*)
- Desnormalizados: $\mathbb{F}_d = \{x \in \mathbb{R} : x = \pm m \times \beta^{L-t}$, con $0 < m < \beta^{t-1}\}$.
  Más pequeño desnormalizado $min_d = \beta^{L-t}$
- $\mathbb{F} = \mathbb{F}_n \cup \mathbb{F}_d \cup \{0\}$. Los elementos de $\mathbb{F}$ no están uniformemente distribuidos: hay huecos!
- **OJO:** El *rango* de números puntos flotantes de $\mathbb{F}$ (distintos de cero) es el conjunto $R(\mathbb{F}) = \{y \in \mathbb{R} : min \leq |y| \leq max\}$.
- Los números en $\mathbb{R}$ que tienen una representación exacta en $\mathbb{F}$ se denominan *números de máquina*

Ejemplo: $\mathbb{F}(\beta, t, L, U) = \mathbb{F}(2, 3, -1, 2)$

- Cardinal de $\mathbb{F}_n$: 16×2 (positivos y negativos)
- $max = (0.111)_2 \times 2^2 = 7/2 \qquad min = (0.100)_2 \times 2^{-1} = 1/4$
- Desnormalizados: $0.010 \times 2^{-1} = 1/8; \quad 0.011 \times 2^{-1} = 3/16;$
  $(0.001)_2 \times 2^{-1} = 2^{-4} = \mathbf{1/16} = min_d$
  Cardinal de $\mathbb{F}_d$: 3×2 (positivos y negativos)
- $\mathbb{F} = \mathbb{F}_n \cup \mathbb{F}_d \cup \{0\} = \{-*, +*\} \cup \{-*, +*\} \cup \{0\}$



- $R(\mathbb{F}) = \{y \in \mathbb{R} : 1/4 \leq |y| \leq 7/2\}$

Aritmética del Computador: $fl(x)$

$fl : R(\mathbb{F}) \to \mathbb{F}$

Supongamos que $x \in R(\mathbb{F})$. El valor $fl(x)$ es la mejor aproximación de x en $\mathbb{F}$

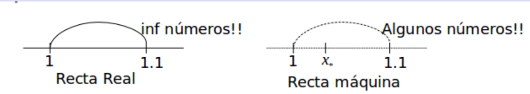
$$fl(x) = (0.d_1 d_2 \ldots \widehat{d_t})_\beta \times \beta^e, \text{ donde } \widehat{d_t} = \begin{cases} d_t & \text{si} \quad d_{t+1} < \beta/2 \\ d_{t+1} & \text{si} \quad d_{t+1} \geq \beta/2 \end{cases}$$

Además (Excepciones):

1. $fl(0) = 0$
2. Cualquier cálculo que produzca un resultado mayor que *max* genera *Overflow* y el resultado es un *punto flotante excepcional* denotado por `Inf`: `Inf + Inf = Inf`.
3. Cualquier cálculo que intente producir un resultado indefinido genera un *punto flotante excepcional* denotado por `NaN` (Not a Number): $0/0 =$ `NaN`, `Inf-Inf = NaN`.
4. Cualquier cálculo que produzca un resultado menor que *min* (min_d) genera *Underflow* y el resultado es 0 [a]

[a] *Cleve Moler: "This involves one of the optional, and controversial, aspects of the IEEE standard. Many, but not all, machines allow exceptional denormal or subnormal floating-point numbers"*

Aritmética del Computador: ϵ_M



PREG.: ¿Quién es el x_* más pequeño tal que el resultado de $1 + x_*$ es mayor que uno?

Epsilon de una máquina con t dígitos de mantisa y base β. (Ejemplo: $t = 5$ y $\beta = 2$)

```
s ← 1; iter ← 0; β ← 2
while 1 + s ≠ 1 do
    s ← s/β
    iter ← iter + 1
end while
```

$$\begin{array}{llcl}
1: & (1+\frac{1}{2})_{10} & \equiv & (1.1)_2 \rightsquigarrow 0.11000 \times 2 \\
2: & (1+\frac{1}{2^2})_{10} & \equiv & (1.01)_2 \rightsquigarrow 0.10100 \times 2 \\
3: & (1+\frac{1}{2^3})_{10} & \equiv & (1.001)_2 \rightsquigarrow 0.10010 \times 2 \\
4: & (1+\frac{1}{2^4})_{10} & \equiv & (1.0001)_2 \rightsquigarrow 0.10001 \times 2 \\
5: & (1+\frac{1}{2^5})_{10} & \equiv & (1.00001)_2 \rightsquigarrow 0.10000 \times 2 = 1
\end{array}$$

RES: x_* es el ***epsilon de la máquina***!

Epsilon de la máquina: Menor valor ϵ_M tal que $1 + \epsilon_M > 1$

- $\epsilon_M = \beta^{1-t}$
- ϵ_M NO es el *cero de la máquina*: El 0 real es un número de máquina.
- ϵ_M NO es el menor valor representable en $\mathbb{F}$: *min* es mucho menor que ϵ_M!

Aritmética del Computador: *fl*, ϵ_m y μ

Si $x \in R(\mathbb{F})$ entonces,

$$\frac{|x - fl(x)|}{|x|} \leq \mu = \begin{cases} \epsilon_M & \text{en caso de} \quad \text{trucamiento} \\ \dfrac{\epsilon_M}{2} & \text{en caso de} \quad \text{redondeo} \end{cases}$$

μ se conoce como la ***unidad de redondeo de la máquina***

Módelo estándar: Sea $x \in R(F)$

$$\begin{aligned} fl(x) &= x(1+\delta) \\ fl(x \circ y) &= (x \circ y)(1+\delta) \end{aligned}$$

con $\delta < \mu$ y $\circ$ se denota cualquiera de la operaciones aritméticas básicas: $+, -, \times, \div$

Cosas raras

1. Las operaciones son conmutativas PERO en general no son asociativas ni distributivas:

$$a := fl(a) = 0.1234567 \times 10^0,\ b := fl(b) = 0.4711325 \times 10^4\ c := fl(c) = -fl(b)$$

CASO 1: $fl(fl(a+b)+c)$:

$$\begin{array}{lll} & 0.00001234567 & \times 10^4 \\ a+b: & 0.4711325 & \times 10^4 \\ \hline & 0.47114484567 & \times 10^4 \end{array}$$

$$r = fl(a+b) = 0.4711448 \times 10^4$$

$$\begin{array}{lll} & 0.4711448 & \times 10^4 \\ r-b: & 0.4711325 & \times 10^4 \\ \hline & 0.0000123 & \times 10^4 \end{array}$$

$$\boxed{0.123 \times 10^0}$$

CASO 2: $fl(a + fl(b+c))$:

$b + c = 0$

$fl(a + fl(b+c)) = a$

$$\boxed{0.1234567 \times 10^0}$$

Cosas raras

2. *Overflow*: $\mathbb{F}(10,4,-4,4)$. En este caso $max = \beta^U(1-\beta^{-t}) = 9999$.

   Si $x_1 = 3457$ y $x_2 = 3$, entonces $x_1x_2 = 10371$, i.e., $x_1x_2 > max$.

   $$\widehat{x}_1\widehat{x}_2 = fl(x_1)fl(x_2) = 0.10371 \times 10^5 \therefore fl(\widehat{x}_1\widehat{x}_2) = \texttt{Inf}$$

3. Falta de precisión: $\mathbb{F}(10,4,-4,4)$

   Si $x_1 = 345,7$ y $x_2 = 3$, entonces $x_1x_2 = 1037,1$

   $$\widehat{x}_1\widehat{x}_2 = fl(x_1)fl(x_2) = 0.10371 \times 10^4 \therefore fl(\widehat{x}_1\widehat{x}_2) = 0.1037 \times 10^4$$

4. *Underflow*: $\mathbb{F}(10,3,-2,3)$. Con $x = (x_1,x_2)^T = (0.01, 0.02)^T$ se quiere calcular $n_x = \|x\|_2$ ($n_x = 0.02236067$)

   $$\begin{array}{lll} \widehat{x}_1 := fl(x_1) = 0.1 \times 10^{-1}; & \widehat{x}_1^2 = 0.1 \times 10^{-3} & \therefore \quad fl(\widehat{x}_1^2) = 0 \\ \widehat{x}_2 := fl(x_2) = 0.2 \times 10^{-1}; & \widehat{x}_2^2 = 0.4 \times 10^{-3} & \therefore \quad fl(\widehat{x}_2^2) = 0 \\ & & \therefore \quad \widehat{n}_x = 0 \end{array}$$

PERO

Cosas raras

4. *Underflow*: $\mathbb{F}(10, 3, -2, 3)$. Con $x = (x_1, x_2)^T = (0.01, 0.02)^T$ se quiere calcular $n_x = \|x\|_2$ ($n_x = 0.02236067$)

Sea $m = \max\{x_1, x_2\} = x_2$. Si $y = (x_1/m, x_2/m)^T$, entonces $\|x\|_2 = m\|y\|_2$

$$
\begin{array}{lll}
\widehat{y}_1 := fl(x_1/m) = \frac{0.1\times 10^{-1}}{0.2\times 10^{-1}} = 0.5 & \therefore & fl(\widehat{y}_1^2) = 0.25 \\
\widehat{y}_2 := fl(x_2/m) = 1 & \therefore & fl(\widehat{y}_2^2) = 1 \\
\|y\|_2 \approx \sqrt{fl(\widehat{y}_1^2) + fl(\widehat{y}_2^2)} = \sqrt{0.125} & \therefore & \widehat{n}_y = 0.112 \times 10 \\
\|x\|_2 \approx m\widehat{n}_y = 0.0224 & \therefore & \widehat{n}_x = 0.224 \times 10^{-1}
\end{array}
$$

MORALEJA: La forma de hacer los cálculos es importante!

Comentarios finales

Algunos mitos

- La resta de dos números parecidos siempre genera problemas: $x+(y-z)$ con $x >> y \approx z > 0$ no genera problemas
- Los errores de redondeo son terribles si muchos de ellos se acumulan luego de una gran cantidad de operaciones: `integral.m`
- Evaluación de ?expresiones cortas? libres de cancelaciones, overflow y underflow son precisas: `idem.m`
- Aumentar la precisión en los cálculos aumenta la precisión de la respuesta: `valorZ.m` ($z = f(2/3) = 1$)
- Los errores de redondeo siempre son malos: `potencia.m`

Bibliografía consultada

- Nicholas J. Higham. Accuracy and Stability of Numerical Algorithms, Second Edition. SIAM 2002.
- Endre Süli and David F. Mayers. An Introduction to Numerical Analysis. Cambridge. 2003.
- Germund Dahlquist and Åke Björck. Numerical Methods in Scientific Computing. SIAM. 2008.
- Alfio Quarteroni, Riccardo Sacco and Fausto Saleri. Numerical Mathematics (Texts in Applied Mathematics 37). Cambridge. 2006.
- David Kincaid and Ward Cheney, Numerical Analysis. Numerical analysis: mathematics of scientific computing. American Mathematical Soc. 2002.