ALN2022
Clase 14
20/10

Estabilidad

En un mundo ideal, los algoritmos en análisis numérico podrían darnos soluciones exactas a nuestros problemas, pero esto dista mucho de la realidad. Los problemas que estamos mirando pertenecen al mundo continuo pero nuestras máquinas digitales viven en un mundo discreto. En este sentido la noción de estabilidad nos da una medida de la precisión de los outputs de nuestros algoritmos.

Recordando el mapa solución $S : I \longrightarrow O$ del espacio de inputs en el espacio de outputs (en general definido localmente como hemos visto) podemos pensar en <u>algoritmo</u> como otro mapa $\tilde{S} : I \longrightarrow O$ que a un input $a \in I$ devuelve una solución aproximada.

Más formalmente, tenemos $S : I \longrightarrow O$, una computadora con la aritmética dada anteriormente (de punto flotante), un algoritmo ~~para~~ y su implementación. Luego dado un input $a \in I$, lo ~~que~~ introducimos en la computadora ~~y~~ resultando en $fl(a)$ (el mapa redondeo) para luego implementar el algoritmo y dar como resultado un nº en punto flotante $\tilde{S}(a)$:

Resumiendo: $a \in I \xrightarrow{\text{computadora}} fl(a) \xrightarrow{\text{algoritmo}} \tilde{S}(a) \in \text{``}\mathbb{F}\text{''}$

Lo anterior de hecho está idealizado, de hecho la misma secuencia de pasos nos podría dar una solución distinta por lo que quizás $\tilde{S}(a)$ puede no estar bien definido.

Pero a pesar de las idealizaciones y complicaciones podemos hacer un análisis riguroso de los algoritmos.

Precisión

Una forma natural de decir que nuestro algoritmo es preciso es que el output dado por el algoritmo esté muy cerca del output verdadero.

En términos ya vistos, decimos que el algoritmo es preciso si

$$\forall a \in I \quad , \quad \frac{\|\tilde{S}(a) - S(a)\|}{\|S(a)\|} = O(\varepsilon_{máquina}).$$

Esta expresión parece natural pero esconde un detalle importante.
¿Qué entendemos por $O(\varepsilon_{máquina})$? ¿Es uniforme en todos los inputs?

Sabemos de la teoría del condicionamiento que el nº de condición es inevitable. Una vez que nuestro input se transforma de a en $fl(a)$ tenemos una pequeña perturbación que puede ser muy significativa si el input está mal condicionado.

Mencionado esto podemos decir que el algoritmo es preciso si

$$\|\tilde{S}(a) - S(a)\| \leq C \cdot \mu(a) \cdot \varepsilon_{máquina} \cdot \|S(a)\|$$ para alguna cte. C que sólo puede depender de las dimensiones.

Estabilidad

La estabilidad de un algoritmo $\tilde{S}$ asociado al mapa S se describe de la siguiente manera. Decimos que $\tilde{S}$ es estable si $\forall a \in I$

$$\frac{\|\tilde{S}(a) - S(\tilde{a})\|}{\|S(\tilde{a})\|} = O(\varepsilon_{máquina}) \quad \text{para } \tilde{a} \in I \text{ tq } \frac{\|a - \tilde{a}\|}{\|a\|} = O(\varepsilon_{máq})$$

En palabras de Trefethen:

"Un algoritmo estable da casi la respuesta exacta a casi la pregunta correcta".

Backward Stability (Estabilidad Inversa)

Una condición más fuerte que la estabilidad es la est. inversa:

$\tilde{S}$ es backward-stable si $\forall a \in I$ $\tilde{S}(a) = S(\tilde{a})$

para $\tilde{a} \in I$ tq $\dfrac{\|\tilde{a}-a\|}{\|a\|} = O(\varepsilon_{maquina})$

i.e "Un algoritmo es b.stable si da la respuesta exacta a casi la pregunta correcta". Observar que si $\tilde{S}$ es b.s. entonces

$$\frac{\|\tilde{S}(a)-S(a)\|}{\|S(a)\|} = \frac{\overbrace{\|\tilde{S}(a)-S(\tilde{a})\|}^{=0} + \|S(\tilde{a})-S(a)\|}{\|S(a)\|} = \frac{\|S(\tilde{a})-S(a)\|}{\|S(a)\|} \leq \mu(a)\cdot\frac{\|\tilde{a}-a\|}{\|a\|} = O(\varepsilon_{mach})$$

Estabilidad a la Smale

Supongamos que estamos en las condiciones vistas anteriormente de que tenemos que la variedad solución $V = \{(a,x) \in I \times O : F(a,x) = 0\}$ con F regular (alcanza $F(a,\cdot)$ sea C^2), y las dimensiones adecuadas.

Luego x es output de a si $F_a(x) = 0$ con $F_a(\cdot) = F(a,\cdot)$.
Se puede construir un mapa de Newton asociado a la función $F_a : O \to \mathbb{R}^k$, y lo definimos por N_{F_a}

Smale define el output de un algoritmo a un output $\tilde{x} = \tilde{S}(a)$ tal que $\tilde{x}$ está en la cuenca de atracción inmediata de convergencia cuadrática del N_{F_a} al cero x.

i.e. $d(N_{F_a}^k(\tilde{x}), x) \leq \left(\frac{1}{2}\right)^{2^k-1} d(\tilde{x}, x)$.

~~Observar que la definición de Smale~~

Smale le da el nombre de "cero aproximado" a los ~~puntos~~ puntos que están en la cuenca de atracción anterior.

Por lo tanto si nuestro algoritmo nos da un cero aproximado, es muy fácil ~~que~~ diseñar un algoritmo estable ya que a nuestro output le podemos aplicar el método de Newton para obtener la precisión que deseamos.

~~Veamos~~ Ejemplos de Estabilidad

Estabilidad de Aritmética de Punto Flotante

Veamos que las operaciones básicas (sus correspondientes algoritmos) son back-estables.

Resta: Sean $(a_1, a_2) \in I^2$ y nuestra función $\delta(a_1,a_2) = a_1 - a_2$

Luego nuestra computadora (con el axioma fundamental) define el algoritmo

$$\tilde{\delta}(a_1, a_2) = fl(a_1) \ominus fl(a_2)$$

i.e. redondea ~~f~~ $a_i \mapsto fl(a_i)$, para luego proceder a realizar un cálculo en $\mathbb{F}$ (que desconocemos) para dar un elemento en $\mathbb{F}$. Recordando el mapa $fl : \mathbb{R} \to \mathbb{F}$ y el axioma se tiene que:

$fl(a_1) = a_1(1+\varepsilon_1)$ $\quad fl(a_2) = a_2(1+\varepsilon_2)$ $\quad$ con $|\varepsilon_1|, |\varepsilon_2| \leq \varepsilon_{mach}$

Luego. $fl(a_1) \ominus fl(a_2) = \big(fl(a_1) - fl(a_2)\big)(1+\varepsilon_3)$

$|\varepsilon_3| \leq \varepsilon_{mach}$.

Por lo tanto

$$\tilde{S}(a_1, a_2) = \left[a_1(1+\varepsilon_1) - a_2(1+\varepsilon_2)\right]\cdot(1+\varepsilon_3)$$

$$= a_1(1+\varepsilon_1)(1+\varepsilon_3) - a_2(1+\varepsilon_2)(1+\varepsilon_3)$$

$$= a_1(1+\varepsilon_4) - a_2(1+\varepsilon_5)$$

(con $|\varepsilon_4| \leq 2\,\varepsilon_{machine} + O(\varepsilon_{mach}^2)$).

y $|\varepsilon_5|$

Hemos concluido que $\tilde{S}(a_1, a_2) = S(\tilde{a}_1, \tilde{a}_2)$

con $\tilde{a}_1 := a_1(1+\varepsilon_4)$, $\tilde{a}_2 := a_2(1+\varepsilon_5)$ satisfaciendo

$$\frac{\|\tilde{a}_1 - a_1\|}{\|a_1\|} = O(\varepsilon_{machine}), \quad \frac{\|\tilde{a}_2 - a_2\|}{\|a_2\|} = O(\varepsilon_{mach})$$

y por lo tanto $\dfrac{\|(\tilde{a}_1, \tilde{a}_2) - (a_1, a_2)\|}{\|(a_1, a_2)\|} = O(\varepsilon_{mach})$

concluyendo que es backward-estable.

Punto FLOTANTE (Revisitado)

[Fijaremos la base $\beta = 2$, y nos permitiremos trabajar con t y s distintos siendo $\underline{t}$ la cantidad de bits para la mantisa y $\underline{s}$ cantidad de bits para el exponente.]

Recordar que un nº real $x \in \mathbb{R}$ se puede escribir en expansión binaria como ~~$x = \pm (d_n 2^n + d_{n-1} 2^{n-1} + \dots +$~~

$$x = \pm \left(d_n 2^n + d_{n-1} 2^{n-1} + \dots + d_0 + d_{-1} 2^{-1} + \dots + \dots \right)$$

donde los d_k son dígitos binarios, i.e. $d_k \in \{0, 1\}$.

Una forma abreviada de lo anterior es escribir

$$x = \pm \left(d_n d_{n-1} \cdots d_0 . d_{-1} d_{-2} \cdots \right)_2 .$$

Un nº en punto flotante lo escribimos como $x = \underbrace{m}_{\text{mantisa}} \times 2^{\overbrace{e}^{\text{expon.}}}$

~~$x = \pm (d_{-1} d_{-2} \dots d_{-t})$~~

donde $m = \pm (b_{-1} b_{-2} \cdots b_{-t})_2$, $e = \pm (c_{s-1} \cdots c_1 c_0)_2$

A diferencia de la clase pasada al exponente lo escribimos explícitamente el exponente. Recordar que $\mathbb{F}$ son los normalizados, es decir $b_{-1} = 1$. ~~Tomando~~ Observar que ahora $\mathbb{F}$ queda bien definida y sólo depende de los parámetros (t, s) y podemos notar el $\mathbb{F} = \mathbb{R}(t, s)$. (Si x no está normalizado casi siempre podemos hacerlo multiplicando 2^{e} ...)

Luego la representación física del punto flotante se modela con los siguientes dígitos

±	b_{-1}	b_{-2}	·	....	b_{-t}	±	c_{s-1}	c_{s-2}	.....	c_1	c_0

$\underbrace{\qquad\qquad}_{t \text{ bits mantisa}}$ $\underbrace{\qquad\qquad}_{s\text{- bits exponente.}}$

El conjunto $\mathbb{R}(t,s)$ es finito. Calculemos explícitamente, en función de los parámetros t, s cuál es el nº mayor, y cuál es el más cercano al 0.

$$\max_{x\in\mathbb{R}(t,s)} |x| = \left(\frac{1}{2}+\frac{1}{2^2}+\dots+\frac{1}{2^t}\right)\times 2^{1+2+2^2+\dots+2^{s-1}} = \boxed{\left(1-\frac{1}{2^t}\right)2^{(2^s-1)}}$$

$$\min_{x\in\mathbb{R}(t,s)} |x| = \frac{1}{2}\times 2^{-(2^s-1)} = \boxed{2^{-2^s}}$$

En 32-bits $\to$ $t=23$, $s=7$ se tiene máx $\cong 2^{2^7} = 2^{128} \simeq 10^{38}$

y mínimo $2^{-128} \simeq 10^{-38}$

En 64-bits ($t=53$, $s=10$) el resultado es 10^{308} y 10^{-308} respectivamente.

Función Redondeo (Revisitado)

La función $fl : \mathbb{R} \to \mathbb{F} = \mathbb{F}(t,s)$ ya la vimos y la idea que opera igual que la función redondeo, que ya conocemos. ~~En nuestro caso es más sencillo~~ ~~dado que hay que~~ Veamos como aplica en el caso binario (que de hecho es más simple)

Sea $x \in \mathbb{R}$, $x = \pm\left(\sum_{k=1}^{+\infty} b_{-k} \cdot 2^{-k}\right) 2^e$

luego cortamos (función chop)

$$fl(x) = x^* \left(= chop(x)\right) = \pm\left(\sum_{k=1}^{t} b_{-k} 2^{-k}\right) 2^e$$

Hay otra posibilidad, que funciona también que es cortar y decidir si lo mandamos a izq o derecha (tipo $10,5 = 11$)
($fl(x) = chop(x + \frac{1}{2} 2^{-t} 2^e)$)

Observar que $$|x^* - x| = \left|\pm \sum_{k=t+1}^{+\infty} b_{-k} 2^{-k}\right| 2^e \leq \left(\sum_{k=t+1}^{+\infty} 2^{-k}\right) 2^e = 2^{-t} \cdot 2^e.$$

Por lo tanto, y como era esperado, el error relativo es

$$\frac{|x^* - x|}{|x|} \leq \frac{2^{-t} \cdot 2^e}{\frac{1}{2} 2^e} = 2^{-t+1}.$$