Divulgación

# Análisis numérico (La nostalgia del continuo)

José Guerrero Grajeda*
e
Irene Sánchez Guevara**

* Facultad de Ciencias, UNAM
** Depto. de Política y Cultura, UAM-X

RESUMEN

Se caracterizan los problemas de interés para el análisis numérico y se analizan las dificultades que se presentan al tratar de resolverlos numéricamente. También se introducen y ejemplifican los conceptos de inestabilidad algorítimica y mal condicionamiento y, finalmente, se dan algunos datos históricos.

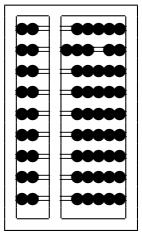
## I. Introducción

En distintos ámbitos de las matemáticas se plantean problemas cuya resolución requiere de efectuar ciertos cálculos numéricos, por ejemplo:

*1*) Dada $f\colon [a,b] \subset R \to R$, calcular $x_* \in [a,b]$ tal que $f(x_*) = 0$.
*2*) Calcular la solución del sistema

$$Ax = b,$$

con $A$ una matriz de elementos reales de orden $n$ y $b \in R^n$.



En las teorías correspondientes, generalmente se tienen resultados acerca de la existencia o no de soluciones, así como de su posible unicidad. Para el caso (1), se sabe que si $f(a)$ y $f(b)$ tienen signos opuestos y $f$ es continua en $[a,b]$, entonces existe solución, aunque no necesariamente única; mientras, para (2), por ejemplo, si $\det(A) \neq 0$, entonces existe solución única.

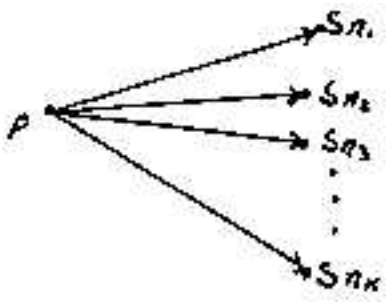
En el presente trabajo trataremos con esta clase de problemas (a la que

denotaremos $C$), cuya estructura numérica básica son los números reales. El cálculo de las soluciones respectivas es uno de los objetivos del análisis numérico o matemática numérica, cuya herramienta fundamental es la computadora digital.

Con respecto al asunto de buscar la solución de un problema determinado en una teoría distinta a la que le dio origen, esto se debe en nuestro caso a varias razones, una de ellas de orden práctico (piénsese por ejemplo en lo que implicaría tratar de resolver (2) para $n = 1000$ contando sólo con papel y lápiz), aunque también existen casos en los que ante la imposibilidad de obtener la solución "analítica" de un problema, se opta por calcular una solución numérica, como es usual en el campo de las ecuaciones integrales y las diferenciales, por ejemplo.

## II. Los costos del cambio

Consideremos ahora $P \in C$ y supongamos que tiene solución única, la cual nos interesa calcular numéricamente. Un problema de origen cuando uno se plantea esta cuestión es que los conceptos de unicidad y existencia en general no se conservan cuando se pasa de la teoría original a la numérica. Gráficamente, un esquema sería:



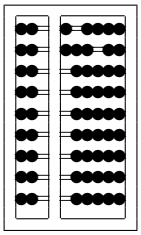

donde $S_p$ es la solución teórica y $S_{n_i}$ es una de las posibles soluciones numéricas, aunque también es factible que no exista solución de este tipo.

La razón fundamental de que esto suceda es que el campo de los números reales es sustituido en la matemática numérica por una estructura conocida como sistema de punto flotante. Este se obtiene a partir de $R$ como sigue:

Sea $x \in R$ de la forma

$$x = \pm\Big(\sum_{k=1}^{\infty} \frac{d_k}{10^k}\Big)10^{e_x},$$

con $d_1 \neq 0$, $0 \leq d_k \leq 9$, $k = 2, 3, \ldots$ y $e_x \in Z$.



Se define ahora la función flotante $fl\colon \Omega \subset R \to R$ como

$$fl(x) = \pm\Big(\sum_{k=1}^{t} \frac{d_k}{10^k}\Big)10^{e_x}, \qquad t \geq 1.$$

A partir de esto se llega a

$$F(\Omega) = \{\tilde{x} \in R \backslash \tilde{x} = fl(x), \quad x \in \Omega\} \cup \{0\}.$$

Este conjunto que es ahora el sustento numérico, tiene entre sus características: *1)* ser discreto y *2)* ser finito. Tenemos pues que el paso a otra teoría conlleva el cambio de lo infinito a lo finito, de lo continuo a lo discreto. El puente entre ambos mundos lo constituiye la función flotante que actúa como una especie de criba que transforma una sección de la recta real en un conjunto de puntos aislados.

En $F(\Omega)$ se definen las operaciones $\oplus$, $\ominus$, $\odot$ y $\oslash$, correspondientes a $+$, $-$, $\cdot$ y $/$ en $R$, por la relación:

$$x_1 \ \circledcirc{\text{op}} \ x_2 = fl(x_1 \,\text{op}\, x_2),$$

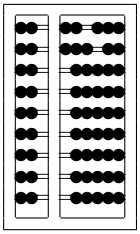
donde $\circledcirc{\text{op}}$ simboliza la operación en $F(\Omega)$ y op la correspondiente en $R$.

Este nuevo sistema presenta diferencias sustanciales con respecto a $R$, debido a la pérdida del continuo. Algunas de ellas son, dados $x_1, x_2, x_3 \in F(\Omega)$

*1)* $x_1 \oplus x_2 = x_1 \not\Rightarrow x_2 = 0$,
*2)* $x_1 \oplus (x_2 \oplus x_3) \neq (x_1 \oplus x_2) \oplus x_3$.

En este ámbito donde es central la realización de cálculos numéricos, cuestiones de este tipo tienen consecuencias tan graves como la ya mencionada pérdida de existencia y unicidad de las soluciones originales; así, por ejemplo, la trivial suma

$$S = \sum_{1}^{m} 1, \qquad m \in N$$



tiene en $F(\Omega)$ múltiples soluciones que dependen esencialmente de la forma como se calcule pues, como ya mencionamos, $\oplus$ no es asociativa. Veamos esto.

En términos llanos, lo que la expresión anterior nos dice es que sumemos la unidad $m$ veces. La teoría matemática nos indica que dicha suma tiene por valor $S = m$.

Pues bien, resulta que a los autores se nos ocurrió proceder como si desconocieramos tal resultado y elaboramos un programa en lenguaje computacional para calcular el valor de la suma en cuestión, trabajando con $m = 10000$ y $F(\Omega)$ con $t = 2$. En términos sencillos, nuestro programa lo que hace es decir a la computadora que ejecute el siguiente:

**Algoritmo 1**.

$S =\leftarrow 0.$
Para $i = 1, 2, \ldots, m,$
$\lfloor S \leftarrow S \oplus 1.$
Terminar.

Lo sorprendente del asunto es que cuando usamos el programa correspondiente en la computadora, para obtener el resultado del proceso completo, éste fue:

$$S = 100,$$

siendo que el resultado (único en $\mathbb{R}$) es

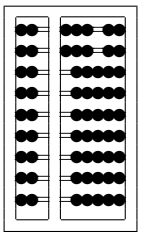
$$S = 10000.$$

Por supuesto, aun desconociendo las teorías sobre los números, es claro que el resultado obtenido es absurdo. La primera impresión que se tiene es que la computadora sólo ejecutó 100 pasos del algoritmo; sin embargo, al revisar el programa checamos que la orden era proseguir hasta $m$. ¿Dónde buscar entonces el error?

Usualmente, cuando un camino no nos lleva al resultado esperado, lo que hacemos es intentar otro. En nuestro caso, modificamos el procedimiento original como sigue:

**Algoritmo 2**.

$S \leftarrow 0.$
Para $\quad i = 1, 2, \ldots, 1000,$
$\quad S_p \leftarrow 0.$
$\quad\quad$ Para $\quad j = 1, 2, \ldots, 10,$
$\quad\quad \lfloor S_p \leftarrow S_p \oplus 1,$
$\quad\quad\quad S \leftarrow S \oplus S_p.$
Terminar.



Como se observa, la diferencia con el primer procedimiento es que ahora se fueron construyendo "paquetes" con valor 10, los cuales fueron sumándose a su vez. Se elaboró el programa computacional correspondiente y esta vez se obtuvo como resultado de su ejecución en la computadora:

$$S = 1000.$$

A pesar de lo absurdo del nuevo valor obtenido, lo que sorprende mayormente es su gran diferencia con respecto al producido mediante el primer algoritmo. ¿A qué se debe la variación entre uno y otro? Tal vez un vistazo a ciertos elementos de la aritmética nos sea de utilidad.

En efecto, si para empezar hacemos uso de la simbología a nuestra disposición, resulta que el proceso de cálculo descrito en el algoritmo 1 lo podemos representar como:

$$\cdots((((((((((1+1)+1)+1)+1)+1)+1)+1)+1)+1)+1)+\cdots,$$

mientras que el correspondiente al algoritmo 2 puede verse así:

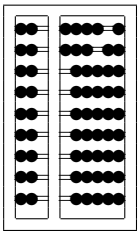
$$\begin{aligned}\{[((((((((((1+1)+1)+1)+1)+1)+1)+1)+1)+1)+1)]+\\ [((((((((((1+1)+1)+1)+1)+1)+1)+1)+1)+1)+1)]\}+\cdots\end{aligned}$$

Ahora, la teoría nos dice que se trata de formas distintas de asociatividad de los términos de nuestra suma original, pero que ambas deben producir exactamente el mismo resultado. En consecuencia, dado que tanto nuestros algoritmos como los correspondientes programas son correctos, esto nos lleva al sorpresivo resultado: *la operación suma en la computadora no cumple la propiedad asociativa.*

Veamos cómo se obtuvieron estos valores de $S$. Como trabajamos con $t = 2$, entonces resulta que:

$$1 = .10 \times 10^1,$$

de donde, procediendo según el primer esquema, resulta:



$S \leftarrow 0;$
$i = 1,$
$\quad S \leftarrow S \oplus .10 \times 10^1 = fl(0 + .10 \times 10^1) = fl(.10 \times 10^1) = .10 \times 10^1;$
$i = 2,$
$\quad S \leftarrow S \oplus .10 \times 10^1 = fl(.10 \times 10^1 + .10 \times 10^1) = fl(.20 \times 10^1) = .20 \times 10^1;$
$\ldots i = 10,$
$\quad S \leftarrow S \oplus .10 \times 10^1 = fl(.90 \times 10^1 + .10 \times 10^1) = fl(.10 \times 10^2) = .10 \times 10^2;$

$i = 11$,
$\quad S \leftarrow S \oplus .10\times 10^1 = fl(.10\times 10^2 + .10\times 10^1) = fl(.11\times 10^2) = .11\times 10^2$;
$\ldots i = 99$,
$\quad S \leftarrow S \oplus .10\times 10^1 = fl(.98\times 10^2 + .10\times 10^1) = fl(.99\times 10^2) = .99\times 10^2$;
$i = 100$,
$\quad S \leftarrow S \oplus .10\times 10^1 = fl(.99\times 10^2 + .10\times 10^1) = fl(.10\times 10^3) = .10\times 10^3$.

Hasta aquí hemos llegado al valor $S = 100$ usando solamente dos lugares para los dígitos. Veamos ahora qué pasa al continuar con el procedimiento:

$i = 101$,
$\quad S \leftarrow S \oplus .10\times 10^1 = fl(.10\times 10^3 + .10\times 10^1) = fl(.101\times 10^3) = .10\times 10^3$.

Como se observa, al aplicar $fl$ a $S + .10 \times 10^1 = .101 \times 10^3$, considerando $t = 2$ volvemos a obtener:

$$S = .10 \times 10^3$$

y como el esquema usado es siempre

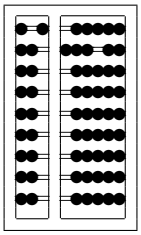
$$S \leftarrow S \oplus .10 \times 10^1,$$

seguiremos obteniendo el mismo valor hasta $i = m$; esto explica el resultado:

$$S_1 = .10 \times 10^3 = 100.$$

Veamos ahora el caso del segundo esquema. Tenemos:

$S \leftarrow 0$;
$i = 1$,
$\quad S_p \leftarrow 0$.
$\quad J = 1$,
$\quad S_p \leftarrow S_p \oplus .10 \times 10^1 = fl(0 + .10 \times 10^1)$
$\qquad\qquad = fl(.10 \times 10^1) = .10 \times 10^1$;
$\quad J = 2$,
$\quad S_p \leftarrow S_p \oplus .10 \times 10^1 = fl(.10 \times 10^1 + .10 \times 10^1)$
$\qquad\qquad = fl(.20 \times 10^1) = .20 \times 10^1$;
$\quad \ldots J = 10$,
$\quad S_p \leftarrow S_p \oplus .10 \times 10^1 = fl(.90 \times 10^1 + .10 \times 10^1)$
$\qquad\qquad = fl(.10 \times 10^2) = .10 \times 10^2$,
$S \leftarrow S \oplus S_p = fl(0 + .10 \times 10^2) = .10 \times 10^2$.



$i = 2$,

$S_p \leftarrow 0.$

$J = 1,$

$S_p \leftarrow S_p \oplus .10 \times 10^1 = fl(0 + .10 \times 10^1) = fl(.10 \times 10^1) = .10 \times 10^1;$

$\ldots J = 10,$

$S_p \leftarrow S_p \oplus .10 \times 10^1 = fl(.90 \times 10^1 + .10 \times 10^1)$

$= fl(.10 \times 10^2) = .10 \times 10^2,$

$S \leftarrow S \oplus S_p = fl(0 + .10 \times 10^2 + .10 \times 10^2)$

$= fl(.20 \times 10^2) = .20 \times 10^2.$

$\ldots i = 10,$

$S_p \leftarrow 0.$

$J = 1,$

$S_p \leftarrow S_p \oplus .10 \times 10^1 = fl(0 + .10 \times 10^1) = fl(.10 \times 10^1) = .10 \times 10^1;$

$\ldots J = 10,$

$S_p \leftarrow S_p \oplus .10 \times 10^1 = fl(.90 \times 10^1 + .10 \times 10^1)$

$= fl(.10 \times 10^2) = .10 \times 10^2,$

$S \leftarrow S \oplus S_p = fl(.90 \times 10^2 + .10 \times 10^2) = fl(.10 \times 10^3) = .10 \times 10^3.$

$i = 11,$

$S_p \leftarrow 0.$

$J = 1,$

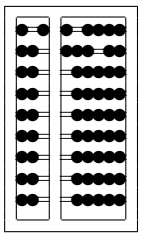
$S_p \leftarrow S_p \oplus .10 \times 10^1 = fl(0 + .10 \times 10^1) = fl(.10 \times 10^1) = .10 \times 10^1;$

$\ldots J = 10,$

$S_p \leftarrow S_p \oplus .10 \times 10^1 = fl(.90 \times 10^1 + .10 \times 10^1)$

$= fl(.10 \times 10^2) = .10 \times 10^2,$

$S \leftarrow S \oplus S_p = fl(.10 \times 10^3 + .10 \times 10^2) = fl(.110 \times 10^3) = .11 \times 10^3.$



Obsérvese cómo, siguiendo este procedimiento, hemos obtenido hasta ahora un valor mayor al valor total obtenido para $S$ con el algoritmo 1.

Continuando con el proceso:

$i = 12$,

$S_p \leftarrow 0.$

$J = 1,$

$S_p \leftarrow S_p \oplus .10 \times 10^1 = fl(0 + .10 \times 10^1) = fl(.10 \times 10^1) = .10 \times 10^1;$

$\ldots J = 10,$

$S_p \leftarrow S_p \oplus .10 \times 10^1 = fl(.90 \times 10^1 + .10 \times 10^1)$

$= fl(.10 \times 10^2) = .10 \times 10^2,$

$S \leftarrow S \oplus S_p = fl(.11 \times 10^3 + .10 \times 10^2) = fl(.120 \times 10^3) = .12 \times 10^3.$

$\ldots i = 100,$

$S_p \leftarrow 0.$

$J = 1,$

$S_p \leftarrow S_p \oplus .10 \times 10^1 = fl(0 + .10 \times 10^1) = fl(.10 \times 10^1) = .10 \times 10^1;$

$\ldots J = 10,$

$S_p \leftarrow S_p \oplus .10 \times 10^1 = fl(.90 \times 10^1 + .10 \times 10^1)$

$= fl(.10 \times 10^2) = .10 \times 10^2,$

$S \leftarrow S \oplus S_p = fl(.99 \times 10^3 + .10 \times 10^2) = fl(.10 \times 10^4) = .10 \times 10^4.$

Tenemos que nuestro segundo algoritmo nos ha llevado al valor:

$$S = .10 \times 10^4 = 1000.$$

Veamos ahora la siguiente iteración:

$i = 101,$

$S_p \leftarrow 0.$

$J = 1,$

$S_p \leftarrow S_p \oplus .10 \times 10^1 = fl(0 + .10 \times 10^1) = fl(.10 \times 10^1) = .10 \times 10^1;$

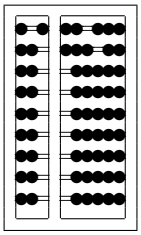
$\ldots J = 10,$

$S_p \leftarrow S_p \oplus .10 \times 10^1 = fl(.90 \times 10^1 + .10 \times 10^1)$

$= fl(.10 \times 10^2) = .10 \times 10^2,$

$S \leftarrow S \oplus S_p = fl(.10 \times 10^4 + .10 \times 10^2)$

$= fl(.1010 \times 10^4) = .10 \times 10^4.$



Como se observa, el valor de $S$ vuelve a ser:

$$S = .10 \times 10^4 = 1000.$$

Por lo que, de continuar el proceso hasta $i = 1000$, obtendremos como resultado final

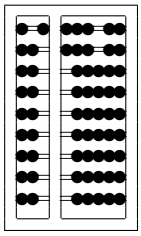
$$S_2 = .10 \times 10^4 = 1000.$$

Este ejemplo, además de mostrarnos la no asociatividad de $\oplus$, nos ha permitido también comprobar cómo un problema que planteado en el ámbito de los números reales tiene solución única, en el campo numérico puede tener múltiples soluciones. Y más aún, nos ha permitido ejemplificar una diferencia importante entre el análisis numérico y las matemáticas de las que éste se nutre: en análisis numérico, el procedimiento (método, algoritmo) elegido para la resolución de un problema es de vital importancia, pues la calidad de la solución obtenida depende estrechamente de él. Resulta entonces natural que uno de los propósitos centrales de quienes trabajan en esta área sea el desarrollo de "buenos" métodos y algoritmos. A este respecto, un buen algoritmo debe plantearse:

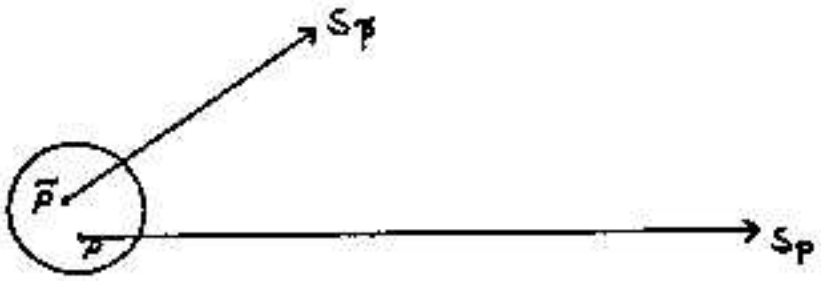
*1*) Optimizar los recursos de cómputo (número de operaciones y uso de memoria).
*2*) Ser numéricamente estable.

La segunda propiedad se cumple cuando es posible mostrar que la solución numérica resulta ser solución exacta de un problema "vecino" del original. En caso contrario, se dirá que el algoritmo es inestable. Gráficamente esto se vería como sigue:


Esto es, $S\tilde{p}$ calculada numéricamente es solución exacta de $\tilde{p}$, con $d(p, \tilde{p})$ pequeña en algún sentido.



Sin embargo, el contar con un buen algoritmo no es condición suficiente para garantizar la correspondiente *bondad* de $S\tilde{p}$, pues bien puede suceder algo del siguiente estilo:




Como se observa, la cercanía de $p$ y $\tilde{p}$ no implica lo mismo para $Sp$ y $S\tilde{p}$. El análisis de este fenómeno ha dado lugar a desarrollos importantes tanto dentro como fuera de la teoría numérica (estudios sobre perturbación y regularización, por ejemplo) los cuales parten del siguiente planteamiento: ¿Qué consecuencias tiene sobre $S_p$ una perturbación pequeña, en principio, de los datos originales de $p$?

Se sabe que una "mala" solución —suponiendo que ha sido calculada mediante un algoritmo adecuado— se debe a las características o condiciones intrínsecas de $P$ y se dice, en este caso, que éste es numéricamente *mal condicionado*. A la subclase de estos problemas pertenecen, entre otros, el de evaluar una función en un intervalo donde su variación es muy grande, y el de calcular la inversa de una matriz cuyas columnas son débilmente independientes. Veamos un ejemplo: consideremos el problema de calcular la solución del sistema:

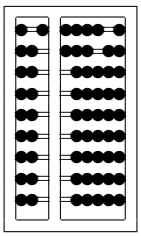
$$1.2969x + 0.8648y = 0.8642$$

$$0.2161x + 0.1441y = 0.1440$$

Un camino sencillo para lograr esto es usar un programa que calcule determinantes.

Como en el ejemplo anterior, recurrimos a una microcomputadora tipo PC compatible y obtuvimos como resultado el siguiente mensaje:

**¡MATRIZ SINGULAR. NO EXISTE SOLUCIÓN O BIEN EXISTE UNA INFINIDAD!**



Sin embargo, resulta que este sistema de ecuaciones tiene como solución única

$$x = 2.0,$$

$$y = -2.0$$

(invitamos al lector a checarlo). Entonces, ¿a qué se debe el error? La teoría nos dice que el uso que hemos hecho de los determinantes es válido,

y por nuestra parte podemos asegurar que nuestro programa es "correcto". ¿Derivará entonces el error del método de solución elegido? Intentemos otro. Uno bastante popular es el que consiste en eliminar una incógnita de una de las ecuaciones y resolver entonces el sistema resultante. Para ello, volvimos a utilizar la microcomputadora y un programa cuya confiabilidad podemos garantizar. Los resultados obtenidos en esta ocasión fueron

$$x = 1.544297,$$

$$y = 1.316604.$$

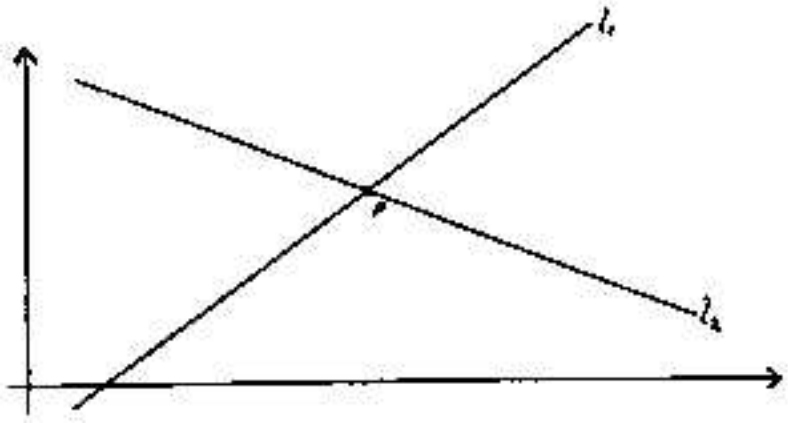
Tal vez podríamos decir que el resultado mejoró, pero aún así, sigue siendo incorrecto. ¿Qué es entonces lo que está pasando? En principio, advertimos que el cambio de método no resolvió el problema, hecho que nos permite aventurar la hipótesis de que nuestra estrategia no es adecuada para este caso. Así, en lugar de continuar aplicando nuevos métodos, reflexionemos un poco sobre el sistema que queremos resolver.

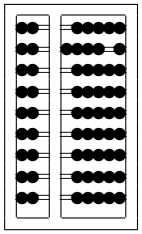
Nosotros sabemos que una interpretación geométrica del cálculo de la solución de un sistema de la forma:

$$a_1x + b_1y = c_1 \tag{1}$$

$$a_2x + b_2y = c_2 \tag{2}$$

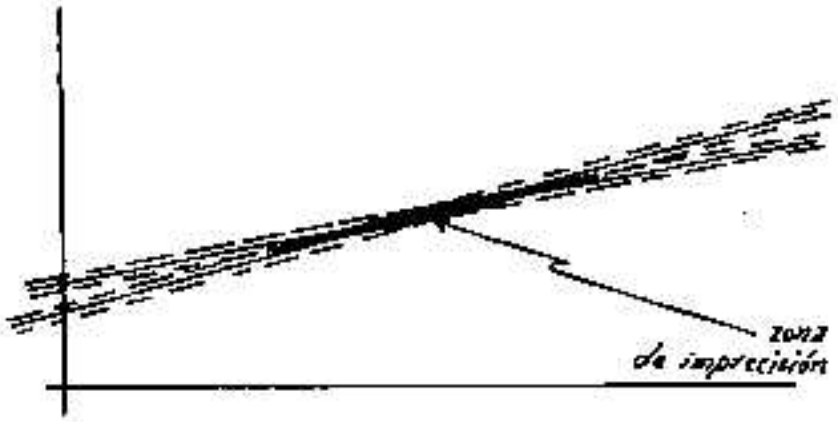
consiste en calcular el punto de intersección de las rectas $l_1$ y $l_2$ que se muestra en la siguiente figura:





donde $l_1$ y $l_2$ son las representaciones gráficas de las ecuaciones (1) y (2). Ahora bien, debido a que la computadora trabaja con aproximaciones numéricas, una interpretación más realista del problema es que cualquier

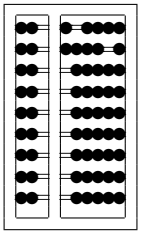
método —por bueno que este sea—, a lo más que puede aspirar es a producir un resultado que esté dentro del paralelogramo mostrado en la siguiente figura:




Según se observa, si $l_1$ y $l_2$ tienden a ser paralelas, es perfectamente posible obtener una solución bastante alejada de la correcta, y ello sin importar qué método utilicemos para su cálculo. Además, cualquier perturbación o imprecisión en los datos —por pequeña que sea—, puede mover el punto de intersección original, lo cual complica aún más la situación. Cuando un caso así se presenta, decimos que estamos ante un sistema mal condicionado. Se invita al lector a checar que nuestro ejemplo es un caso particular en el que se presenta este tipo de mal.

Con respecto a la cuestión de cómo se puede detectar en general el mal condicionamiento de un sistema de $n$ ecuaciones lineales con $n$ incógnitas, se trata de un tema que cae fuera de los alcances del presente trabajo; pero lo que sí podemos decir es que toda buena subrutina para resolver tal tipo de sistemas, debe incluir el cálculo de un parámetro numérico conocido como número de condición o condicional de $A$, simbolizado $K(A)$, con la propiedad $K(A) \geq 1$, y que funciona, en términos generales, de la siguiente manera:

*1*) Si $K(A) \approx 1$, el sistema es bien condicionado.
*2*) Si $K(A) \gg 1$, el sistema es mal condicionado.



La subrutina de eliminación usada para nuestro ejemplo, nos reportó

$$K(A) = 4.673 \times 10^8$$

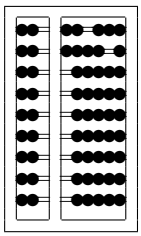
que, como vimos, resultó demasiado para una microcomputadora trabajando en precisión simple.

## III. Condiciones para una teoría

De lo expuesto hasta ahora, resulta que los analistas numéricos están interesados, entre otras cosas, en el cálculo de ciertos entes cuya existencia y unicidad pertenecen a un universo infinto y continuo del que sólo queda la nostalgia. El paso a lo discreto tiene el efecto de trasladar la solución buscada a una especie de galería de espejos en la que el especialista debe aventurarse con sumo cuidado, armado de elementos que le permitan distinguir con claridad una imagen de su original, cosa nada fácil en ocasiones. Asimismo, ante la imposibilidad de obtener en general una solución exacta que parece escaparse siempre, nuestros expertos han de conformarse con una aproximación razonablemente cercana, de donde podría decirse que un analista numérico considera exitoso su tratamiento de un problema si es capaz de ubicarlo en un esquema del siguiente estilo:

esto es, si dispone de hipótesis que le permitan afirmar que partiendo de $\tilde{p}$ "cercano" a $p$, $S\tilde{p}$ resulta "cercana" a $Sp$. Lamentablemente esta no es la situación en la mayoría de los casos, dado el estado actual de la teoría numérica, y la razón fundamental de ello es que, como ya se ha mencionado, en esta disciplina los costos son de vital importancia, razón por la cual un resultado teórico de tipo general para un problema dado puede resultar poco útil en términos prácticos. Por ahora, los especialistas están más preocupados por el desarrollo de nuevos métodos que compitan ventajosamente con los ya existentes, así como con la mejora de éstos y en la justificación de sus resultados.

Se vive pues un proceso que es más de dispersión que de síntesis, aunque existen ya trabajos teóricos importantes de este tipo sobre todo en el área de álgebra lineal. Habrá que esperar, sin embargo, algún tiempo para que toda esa experiencia acumulada hasta ahora de lugar a teorías que permitan superar la actual etapa de "empirismo" en que se encuentra la matemática numérica.



## IV. Notas históricas

Las cuestiones a que nos hemos referido en el presente trabajo surgen y comienzan a ser estudiadas desde el mismo inicio del análisis numérico como una disciplina autónoma, cosa que ocurre durante los años cuarentas; su

nacimiento se liga para algunos con la aparición del trabajo: *Numerical Inverting of Matrices of High Order*, en 1947, en el que sus autores H. H. Goldstine y J. von Neumann llaman la atención de los usuarios acerca de fenómenos como la inestabilidad algorítimica y el mal condicionamiento. Sin embargo, es J. H. Wilkinson quien llega a establecer una serie de resultados clásicos relacionados con los efectos de redondeo numérico en una importante variedad de algoritmos, particularmente en las áreas del álgebra lineal y el cálculo de ceros de polinomios. Es él quien lleva a su expresión más acabada la metodología conocida como análisis retrospectivo del error, cuyos iniciadores fueron C. Lanczos y W. Givens, que en términos generales consiste en determinar si una solución numérica, digamos $\tilde{S}_p$ del problema $p$, calculada usando un algoritmo $A$, resulta ser la solución de $\tilde{p}$ calculada en aritmética exacta, donde éste difiere del original en que sus datos iniciales son los de $p$ más una pequeña perturbación.

Resuelto esto, se obtiene la clasificación de los algoritmos en estables e inestables, comentada en una sección anterior y que resulta de gran importancia en el trabajo numérico. De igual forma, sus estudios sobre la sensibilidad de las raíces de un polinomio debida a pequeñas perturbaciones en sus coeficientes, llevaron pronto a abandonar el cálculo de valores propios de matrices usando el polinomio característico y, de su interacción con C. Reinsch, surgirá el conocido *Handbook for Automatic Computation*, considerado como una contribución fundamental al desarrollo del *software* matemático.

Para los interesados en lo hasta aquí tratado y que deseen ampliar su visión sobre algún punto particular, se da una bibliografía básica.

## Referencias

[1] Danhlquist, G. y A. Bjorck, *Numerical Methods*, Prentice-Hall, Inc., EUA, 1974.

[2] Guerrero, J., "El cálculo científico por computadora: dos fuentes importantes de error", en *Memorias del Congreso Nacional: pasado, presente y futuro de la computación*, UNAM, México, 1988.

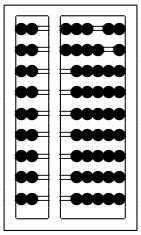
[3] Kahaner, D., C. Moler y S. Nash, *Numerical Methods and Software*, Prentice-Hall, Inc., EUA, 1989.

[4] Neumann, J. von, "Solution of Linear Systems of High Order", en *Collected Works V*, 1963.

[5] ——, "Numerical Inverting of Matrices of High Order", en *Collected Works V*, 1963.

[6] Wilkinson, J. H., *Rounding Errors in Algebraic Processes*, Prentice-Hall, Inc., EUA, 1964

[7] ——, *The Algebraic Eigenvalue Problem*, Oxford University Press, 1965.



[8] ——, "Modern Error Analysis", *SIAM Rev.* **14**, 1971.
[9] ——, "The Perfidious Polinomial", en *Studies in Numerical Analysis*. G. Golub (Editor), Mathematics Asociation of America, 1984.
[10] Wilkinson, J. H. y C. Reinsch (Editores), *Handbook for Automatic Computation, Volume II: Linear Algebra*, Springer-Verlag, 1971.