Publicaciones Matemáticas del Uruguay
Volumen 17, Julio 2019, Páginas 109–118
S 0797-1443

COMPLEJIDAD EN ANÁLISIS NUMÉRICO Y EL PROBLEMA DE VALORES PROPIOS

DIEGO ARMENTANO

Abstract. La complejidad de un algoritmo es el número de pasos requeridos para pasar de una entrada a una salida. El conocimiento de este número nos permite comparar distintos métodos y poder determinar cuáles son más eficientes. Sin embargo el estudio de la complejidad es un tema muy complicado del cuál se sabe muy poco, aún en problemas básicos como encontrar raíces de polinomios o valores propios de matrices. En esta artículo daremos un paseo por distintos problemas y métodos, discutiendo en cada caso qué se sabe (o no se sabe) sobre la complejidad. (La exposición del artículo sigue la charla dictada por del autor para el 6to Coloquio Uruguayo de Matemática.)

1. Introducción

El *análisis numérico* se ocupa del diseño y estudio de *algoritmos* diseñados para resolver problemas clásicos de análisis. Algunos ejemplos clásicos de estos problemas son: interpolación de funciones, integración y diferenciación numérica, aproximación de soluciones de sistemas de ecuaciones y de ecuaciones diferenciales, optimización, etc.

Por algoritmo entendemos una secuencia de instrucciones que son ejecutadas a partir de una entrada (*input*), la cual eventualmente producirá una salida (*output*).

Existen dos clases muy diferenciadas de algoritmos en análisis numérico. Los más sencillos son los algoritmos denominados *métodos directos* que funcionan en una cantidad acotada de pasos. Ejemplos de este tipo son: la eliminación gaussiana, o cualquier problema en el que exista una solución con fórmula explícita (e implementable).

Por el otro lado están los algoritmos *indirectos* que consisten en un sistema dinámico que aproxima a la solución buscada, pero que requiere (aún en un modelo sin errores) infinitos pasos para dar con la solución exacta. Para estos casos es necesario dar una condición de parada a nuestro algoritmo, como por ejemplo que el output esté suficientemente cerca de la solución exacta (ver Sección 3.1). Un ejemplo fundamental, como consecuencia de los trabajos de Abel y Galois, es el de encontrar raíces de polinomios con grado mayor a 4. Los métodos indirectos son los más complicados e interesantes de estudiar como veremos a continuación.

2. Condicionamiento

La primer dificultad para el análisis de un algoritmo son los *errores de redondeo*. Las computadoras actuales trabajan con el sistema de aritmética de *punto flotante*, y entonces todas las operaciones aritméticas son realizadas en un subconjunto finito $\mathscr{E}$ de los números reales $\mathbb{R}$. Unas de las características de este sistema aritmético es

la existencia de una función $r : \mathbb{R} \to \mathscr{E}$ llamada *función de redondeo* que aproxima el dato ingresado (en algún sentido le da el valor del elemento más próximo de $\mathscr{E}$). De esta manera todas las operaciones aritméticas en $\mathbb{R}$ pasan a operaciones aritméticas en $\mathscr{E}$ a través de la función r.

En general, como se comentó en el punto anterior, nuestros datos y operaciones aritmética están sujetos a pequeños errores de redondeos. Por lo tanto, dado un algoritmo que intente resolver nuestro problema, el resultado será una solución de un problema perturbado. Existe una cantidad denominada *número de condición* que cuantifica qué tan sensible es nuestro problema a pequeñas perturbaciones en la entrada. (Es importante destacar que el condicionamiento es una característica intrínseca del problema que estamos mirando, y no depende del algoritmo en cuestión.)

A continuación veremos un ejemplo que ilustra las dificultades mencionadas.

Ejemplo 2.1. Supongamos que queremos resolver el sistema de ecuaciones lineales

$$Ax = b, \tag{2.1}$$

donde $A \in \mathbb{R}^{m\times m}$ y $b \in \mathbb{R}^m$ son las entradas. Al ingresar estas entradas a la computadora obtenemos que A y b son sustituidos por

$$\widehat{A} := A + \Delta A, \quad \widehat{b} := b + \Delta b,$$

donde ΔA y Δb son errores producidos por la aritmética de la computadora. (Para simplificar la exposición supongamos que no hay error de aproximación en b, i.e. $\Delta b = 0$.)

Luego, aún en el caso de estar en un modelo ideal de algoritmo que funcione sin errores, tenemos que nuestro output tiene la forma

$$\widehat{x} := x + \Delta x := (\widehat{A})^{-1} b,$$

siendo $x := A^{-1}b$ la verdadera solución a nuestro problema.

El número de condición, asociado al input A, controla el tamaño del vector Δx. Se prueba que

$$\frac{\|\Delta x\|}{\|x\|} \leq \kappa(A) \frac{\|\Delta A\|}{\|A\|},$$

siendo $\kappa(A) = \|A\| \cdot \|A^{-1}\|$ el número de condición asociado a la entrada A, y $\|\cdot\|$ la norma euclideana. (Ver por ejemplo Blum et al. [9, Capítulo 11].)

En particular esto sugiere que un número de condición grande conspira contra la precisión de nuestro output. (Observar que en este ejemplo no hemos especificado ningún algoritmo.)

El número de condición tiene asociado una interpretación geométrica importante que viene dada por el *Teorema de número de condición.* Si sobre el espacio de matrices $\mathbb{R}^{n\times n}$ consideramos la norma de *Frobenius*, $\|A\|_F := \sqrt{\sum_{i,j} a_{ij}^2}$, entonces se tiene el siguiente resultado.

Teorema 1 (Eckart-Young). *Sea $A \in \mathbb{R}^{n\times n}$ invertible. Entonces*

$$\kappa(A) = \frac{\|A\|}{\operatorname{dist}_F(A, \Sigma)}$$

siendo $\Sigma \subset \mathbb{R}^{n\times n}$ el conjunto de matrices de determinante cero, y d_F la distancia inducida por la norma de Frobenius. (Ver [9, página 202]*.)*

El conjunto Σ es exactamente el conjunto de entradas problemáticas dado que el sistema no tiene solución. (En el contexto del problema de resolver sistemas lineales, Σ es el conjunto de *entradas mal condicionadas*.)

El teorema anterior dice que el número de condición es inversamente proporcional a la distancia de la matriz al conjunto de matrices no invertibles.

Observación 2.2. El teorema de número de condición vale en muchos contextos, por ejemplo ver Shub-Smale [24] para el caso de ceros de polinomios, Armentano [1] para el problema de valores propios, Cucker et al. [15] para el problema de conteo de raíces reales. Hasta ahora no hay una teoría general sobre el número de condición en problemas computacionales. En cada caso particular se estudia si hay un resultado similar.

3. Complejidad algorítmica

3.1. **Ceros aproximados.** Una pregunta importante es saber cuándo nuestro algoritmo debe parar, y para esto es necesario establecer un criterio.

En el siguiente párrafo describiremos dos criterios diferentes.

- *ε-aproximación:* Fijado un $\varepsilon > 0$ de antemano, una *ε-aproximación* es un output que está a una distancia menor o igual a ε de una solución verdadera.
- *Aproximación à la Smale:* Un output es un *cero aproximado* en el sentido de Smale si una versión apropiada del método de Newton comenzando en él converge cuadraticamente, de manera inmediata, a una solución verdadera.

Observar que ser un cero aproximado es más fuerte que ser una ε-aproximación. Esto resulta fácilmente del hecho de que podemos convertir un cero aproximado en una ε-aproximación simplemente iterando una cantidad prefijada (que depende de ε) de veces el método de Newton.

En el otro sentido, no es verdad que toda ε-aproximación puede ser convertida en un cero aproximado. Por ejemplo, con el ε prefijado, es posible que existan más de una solución en el entorno de tamaño ε. O peor aún, la solución puede tener "multiplicidad" y en tal caso el método de Newton converge de manera lineal (ver [9]).

3.2. **Complejidad media.** Una buena medida de la eficiencia de un algoritmo es su *complejidad*. *Grosso modo* la complejidad de un algoritmo, diseñado para resolver un problema de análisis numérico, es el *tiempo* requerido por el mismo para pasar de la entrada a la salida. Por tiempo se entiende la cantidad de operaciones aritméticas básicas (sumas, multiplicaciones, inversiones, etc.) realizadas por el algoritmo.

La complejidad de un algoritmo depende de la entrada que estemos trabajando. Por ejemplo, es claro de la discusión anterior sobre el número de condición que una entrada mal condicionada lleve más tiempo que una entrada bien condicionada. En este sentido la complejidad de un algoritmo depende fuertemente de la entrada que estemos mirando.[1]

Para evitar la dependencia en la entrada, Smale [29] propone una manera sistemática de analizar la complejidad de un algoritmo considerando un análisis probabilístico del mismo. Si consideramos una medida de probabilidad ν sobre el

[1]Además también depende de la solución asociada a nuestra salida. Esto se puede ver claramente en el caso de encontrar raíces de un polinomio donde hay diferentes soluciones y algunas pueden presentar más dificultades que otras para localizarlas.

espacio de entradas $\mathfrak{I}$, ¿cuál es el valor esperado de la complejidad? Este número es lo que se define por *complejidad media*. Conocer la complejidad media nos permite, no sólo comparar algoritmos, sino poder decidir si el problema computacional tiene *complejidad media polinomial*. Con esto nos referimos a problemas tales que existe un algoritmo diseñado para resolverlo con complejidad K, donde su promedio sobre todas las entradas satisface

$$\int_{a\in\mathfrak{I}} K(a)\, d\nu(a) \leq C(\dim \mathfrak{I})^q,$$

donde $\dim \mathfrak{I}$ es la dimensión espacio del espacio de entradas, y q un natural.

3.3. **Complejidad del teorema de Bézout.** En las últimas décadas, motivado por lo trabajos pioneros Shub-Smale [24, 25, 26, 27], se han realizados muchos trabajos con el objetivo de diseñar algoritmos –para el problema de resolver sistemas de ecuaciones– con complejidad media polinomial. El problema 17 de la lista de problemas para el siglo XXI de Steve Smale dice lo siguiente:

> *"Can a zero of n complex polynomial equations in n unknowns be found approximately, on the average, in polynomial time with a uniform algorithm?"* S. Smale [31].

Una respuesta afirmativa a esta pregunta se obtuvo en 2017 por P. Lairez [19], mediante una derandomización del algoritmo randomizado con complejidad media polinomial propuesto por Beltrán-Pardo [7]. (Ver también Beltrán-Pardo[8] y Bürgisser-Cucker [10].)

4. Complejidad del problemas de valores propios

En esta sección daremos un breve paseo sobre la complejidad del problema de valores propios.

El problema de valores propios es el siguiente: dada una matriz A, de tamaño $m \times m$, encontrar un escalar λ y un vector no nulo v tales que

$$(\lambda \mathsf{Id}_m - A)v = 0. \tag{4.1}$$

Nos restringiremos a trabajar sobre el cuerpo de los reales o complejos. Además, observar que el sistema (4.1) es homogéneo en v y por lo tanto podemos trabajar sobre el espacio proyectivo asociado.

Este es un problema común para los científicos que día a día "resuelven" este problema mediante una computadora. Sin embargo, el problema de estudiar su complejidad estuvo abierto por más de 30 años, convirtiéndolo en un de los grandes desafíos para la comunidad de álgebra lineal numérica.

> *"So the problem of devising an algorithm [for the eigenvalue problem] that is numerically stable and globally (and quickly!) convergent remains open."* J. Demmel [14, page 139]

En lo que sigue comentaremos de manera sucinta qué se sabe sobre la complejidad de los algoritmos más conocidos para el problema de valores propios.

4.1. **Polinomio característico.** Es probable que el primer algoritmo que se nos ocurra para atacar el problema de encontrar valores propios de una matriz A, es calcular las raíces del polinomio característico χ_A, para luego encontrar (aproximar) sus raíces. Sin embargo este método parece no ser el apropiado debido a que es

numéricamente inestable como veremos a continuación. El polinomio de Wilkinson (de aspecto inofensivo)

$$W(x) := \prod_{i=1}^{20}(x-i) = x^{20} + w_{19}x^{19} + \cdots + w_1 x + w_0$$

se utiliza como ejemplo de este fenómeno. Supongamos que A es la matriz diagonal con entradas $1, 2, \ldots, 20$ (y por lo tanto $\chi_A(x) = W(x)$). Luego un error del orden 2^{-23} en el coeficiente $w_{19} = -210$ produce, aún si los demás coeficientes son calculados de manera exacta, una variación enorme en los ceros de χ_D. Por ejemplo, el cero 18 y el cero 19 se unen para crear una raíz doble en 18.62, la cual se convertirá en dos raíces complejas conjugadas si el error es aún mayor. (En otras palabras, esto muestra que el polinomio $W(x)$ está mal condicionado en la raíces 18 y 19.)

Sin embargo, se puede probar que la matriz A está bien condicionada para el problema de valores propios (cf. Sección 2), i.e., una pequeña perturbación en las entradas de la matriz A provocan una pequeña alteración en los valores propios. (Ver por ejemplo Armentano [1, Sección 3]).

Observación 4.1. El fenómeno como el que ocurre con el polinomio de Wilkinson motivó a los analistas numéricos a descartar este procedimiento para el problema de valores propios. Más aún, la comunidad no sólo descarta este método para atacar el problema sino que por el contrario, para encontrar raíces de polinomios se transforma el problema a encontrar valores propios de una matriz con las raíces del polinomio, conocida en la literatura como la *matriz compañera*. (Ver por ejemplo Trefethen-Bau [32].) (En Bürgisser-Cucker-Rocha [12] se da una posible explicación del porqué de este fenómeno.)

4.2. **Método de la potencia.** Dada una matriz A, el *método de la potencia* se define por la iteración

$$x_k := \frac{Ax_{k-1}}{\|Ax_{k-1}\|}, \quad (k = 1, 2, \ldots). \tag{4.2}$$

donde x_0 es cierto vector unitario inicial. (Observar que el método está definido siempre que x_k no esté en el núcleo de A).

El método de la potencia es un algoritmo sencillo para encontrar el vector propio dominante. Su simplicidad, desde el punto de vista de sistemas dinámicos, lo hace muy atractivo. Sin embargo este método tiene un lado negativo: no es eficiente desde el punto de vista del análisis numérico, y además sólo encuentra un vector propio.

Si existe un valor propio dominante, entonces es un ejercicio verificar que para casi todo vector inicial la sucesión 4.2 converge a una dirección dominante.

Kostlan [17] obtuvo los primeros resultados finos sobre la complejidad de este método para encontrar ε-aproximaciones del vector propio dominante. En particular se tiene el siguiente resultado.

Teorema 2 ([17])**.** *Sea A una matriz aleatoria $m \times m$ simétrica con entradas independientes gaussianas estándar fuera de la diagonal, y gaussianas con varianza 2 en la diagonal. Entonces la complejidad media del método de la potencia es infinito.*

La randomización elegida resulta de la distribución *gaussian orthogonal ensamble* (GOE) en matrices simétricas $m \times m$, siendo esta la única medida de probabilidad invariante bajo la acción por conjugación del grupo ortogonal.

Si uno excluye un entorno tubular de las matrices mal condicionadas para este problema, o refina el método de la potencia, puede obtener resultados de complejidad media finita para aproximar el vector propio dominante. Ver Kostlan [17] y [18].

4.3. **Iteración de cociente Rayleigh.** Un método más eficiente es la *iteración cociente de Rayleigh* que consiste en: dado un vector inicial unitario x_0, se genera una sucesión $\{x_k\}$ dada por

$$\lambda := x_{k-1}^T A x_{k-1}, \quad (A - \lambda \mathsf{Id})y = x_{k-1}, \quad x_k := \frac{y}{\|y\|}, \qquad (k = 1, 2, \ldots). \tag{4.3}$$

La idea detrás de este método está en usar el método de la potencia para la matriz $(A - \lambda \mathsf{Id})^{-1}$ cuando λ está cerca de ser un valor propio, para luego actualizarlo.

Es conocido que este método es convergente (y rápido) para el caso de matrices simétricas. (Ver por ejemplo Ostrowski [20], Parlett-Kahan [21], y Batterson-Smillie [5]

Sin embargo, no existen resultados relacionados a la complejidad media del mismo.

En el caso de matrices no simétricas, Batterson-Smillie [6] probaron que la situación es muy distinta dado que no existe convergencia global.

Teorema 3 ([6])**.** *Si $m \geq 3$, entonces hay un abierto de matrices para las cuales cada una tiene una abierto de condiciones iniciales donde el método iterativo del cociente de Rayleigh no converge.*

Es claro que resulta entonces que la complejidad media es ∞.

4.4. **Algoritmo QR.** El algoritmo QR, desarrollado por J. Francis en la década del 50, es el método más conocido para encontrar valores propios de una matriz. Este método se basa en la descomposición QR en la cual toda matriz A se puede factorizar de la forma $A = QR$, donde Q es ortogonal y R triangular superior (análogo en el caso complejo).

El algoritmo QR genera una sucesión $\{A_k\}$ que se construye de la siguiente manera: con matrices iniciales de $A_0 := A = Q_0 R_0$, definimos

$$A_k := R_{k-1} Q_{k-1} = Q_k R_k, \qquad (k = 1, 2, \ldots).$$

Es un ejercicio verificar que las matrices A_k son ortogonalmente conjugadas a A, y por lo tanto sus espectros coinciden.

Resulta que en condiciones genéricas A_k tiende a una matriz triangular superior, y por lo tanto en la diagonal están sus valores propios. (Ver Dedieu [13].)

Se puede probar que el primer vector de la matriz A_k coincide (a menos de un escalar) con la sucesión (4.2) generada por el método de la potencia para la matriz A. Por lo tanto, en el caso simétrico, resulta del Teorema 2 que la complejidad media del algoritmo QR para el caso GOE es infinito.

Por esta razón se utilizan variantes del método QR, denominados algoritmos QR con shift, los cuales potencian la velocidad trasladando la matriz A por múltiplos de la identidad (cf. (4.3).)

Actualmente el método más utilizado es al algoritmo QR de Francis con shift (dobles) (ver Batterson [4]). Sin embargo no hay teoremas que aseguren su convergencia en casos no estructurados, y menos aún un estudio de la complejidad media del mismo.

4.5. **Flujos isoespectrales.** Otro tipo de algoritmos son los asociados a flujos isoespectrales. Sobre el espacio de matrices simétricas se considera un flujo ϕ_t que satisface las siguientes propiedades:

- (isoespectral) espectro de $\phi_t(A)$ es constante, $t \geq 0$;
- (convergencia) el flujo $\phi_t(A)$ converge a una matriz diagonal cuando $t \to +\infty$.

Un ejemplo de esto es el flujo generado por la ecuación diferencial de corchete doble de Brockett

$$\dot{A}(t) = [A(t), [A(t), D]], \quad A(0) = A, \quad D = \operatorname{diag}(1, 2, \ldots, m).$$

(Ver [16, Capitulo 2].)

Discretizaciones de estos algoritmos inducen algoritmos para resolver el problema de valores propios. Sin embargo, no existen resultados de complejidad media para estos métodos.

Sería interesante tener resultados que vincularan la longitud de las lineas de flujos con la complejidad de estos algoritmos.

4.6. **Métodos de homotopía.** En esta sección desarrollaremos algoritmos de continuación, o llamados también *métodos de homotopía*, orientado para el problema de valores propios de matrices complejas.

(Nos concentraremos en el problema de valores propios, pero se destaca que *mutatis mutandis* se puede extender a otros problemas numéricos.)

Brevemente, los métodos de homotopía pueden describirse de la siguiente manera. El sistema de ecuaciones $(\lambda I_m - A)v = 0$, con $v \in \mathbb{P}(\mathbb{C}^m)$, siendo $\mathbb{P}$ el espacio proyectivo asociado, es el punto final de un camino de problemas

$$(\lambda(t)I_n - A(t))v(t) = 0, \ v(t) \neq 0, \ 0 \leq t \leq 1, \tag{4.4}$$

con $(A(1), \lambda(1), v(1)) = (A, \lambda, v)$.

Comenzando desde una terna $(A(0), \lambda(0), v(0))$ "continuamos" este camino hasta el sistema objetivo $(\lambda I_n - A)v = 0$. La manera algorítmica de hacerlo es construir un número finito de ternas

$$(A_k, \lambda_k, v_k), \quad 0 \leq k \leq K, \tag{4.5}$$

con $A_k := A(t_k)$, y $0 = t_0 < t_1 < \cdots < t_K = 1$, y donde (λ_k, v_k) es la aproximación del valor y vector propio $(\lambda(t_k), v(t_k))$ de A_k.

Las ternas dadas en (4.5) se construyen utilizando un método de *predicción-corrección* de la siguiente manera: dado la partición $0 = t_0 < t_1 < \cdots < t_K = 1$, y el par $(\lambda_0, v_0) \in \mathbb{C} \times \mathbb{P}(\mathbb{C}^m)$, se define

$$(\lambda_k, v_k) := N_{A_k}(\lambda_{k-1}, v_{k-1}), \quad 1 \leq k \leq K,$$

donde N_A es el operador de Newton N_A definido en $\mathbb{C} \times \mathbb{P}(\mathbb{C}^m)$ (ver Armentano [1]).

La partición $0 = t_0 < t_1 < \cdots < t_K = 1$ se construye de tal manera que el par (λ_k, v_k) es siempre un cero aproximado de A_k.

La complejidad de este algoritmo es el número K de pasos suficientes para validar esta aproximación.

Un resultado importante para los métodos de homotopía es vincular la complejidad K con alguna propiedad analítico geométrica del problema.

Teorema 4 ([1]). *Para $t \in [0,1]$, sea $\Gamma(t) = (A(t), \lambda(t), v(t))$ satisfaciendo (4.4), y $\|A(t)\| = 1$. Entonces, la complejidad K satisface*

$$K \leq \int_0^1 \left\|\dot{\Gamma}(t)\right\| \mu\left(\Gamma(t)\right) \, dt,$$

siendo $\left\|\dot{\Gamma}(t)\right\|$ la velocidad de la curva en $\mathbb{C}^{m\times m} \times \mathbb{C} \times \mathbb{P}(\mathbb{C}^m)$, y μ el número de condición que está dado por

$$\mu(A, \lambda, v) = \|A\|_F \cdot \left\|\left(\Pi_{v^\perp}(\lambda \mathsf{Id} - A)|_{v^\perp}\right)^{-1}\right\|,$$

donde denotamos por $\Pi_{v^\perp}$ es la proyección ortogonal sobre el ortogonal a v.

Este resultado fue motivado por Shub [23], que es el resultado análogo para el caso de la complejidad del problema de Bézout.

El Teorema 4 motiva dos preguntas importantes.

- ¿Cómo elegir la terna inicial (A_0, λ_0, v_0) satisfaciendo (4.1)?;
- ¿Cómo conocer *a priori* caminos $\{A(t)\}$ tales que el levantado $\Gamma(t)$ satisface $\mu(\Gamma(t))$ es pequeño?

La primer pregunta tiene fácil respuesta, dado que hay muchas formas distintas de tomar ternas iniciales. Aunque es importante destacar que cada elección influenciará la complejidad, y por lo tanto es razonable preguntarse si hay una terna óptima. (Ver Problema 7 de la lista de Smale [31].)

Para responder la segunda pregunta vale la pena mencionar que en este caso también hay un teorema de número de condición (cf. Observación 2.2). Por lo tanto basta considerar curvas que se mantengan alejadas de los problemas mal condicionados, i.e., matrices con valores propios múltiples (ver [1]). Dado que no se conoce bien la geometría del problema, tener una respuesta a la segunda pregunta es de suma dificultad. En general se opta por tomar segmentos de recta, o realizar caminos que preserven el condicionamiento. Esto último es viable dado que en general el número de condición tiene ciertos invariantes conocidos *a priori*, como por ejemplo en este caso, el número de condición es invariante por conjugaciones por matrices unitarias.

El primer resultado sobre complejidad para este problema, para el caso de matrices hermitianas, se encuentra en Armentano-Cucker [3] donde los autores dan un algoritmo randomizado con complejidad media polinomial.

El primer resultado satisfactorio sobre la complejidad del problemas de valores propios es reciente, y es el siguiente.

Teorema 5 ([2]). *Hay algoritmos de homotopía, tales que si el input es una matriz A compleja $m \times m$, con entradas gaussianas estándar independientes, entonces el output es un cero aproximado de la matriz A con complejidad media polinomial en m.*

Las cotas de la complejidad media para encontrar un valor y vector propio de matrices complejas gaussianas son de orden $O(m^7)$. Este resultado es una cota superior, pero es posible que pueda ser mejorada.

En el artículo [2] se detallan cuáles son los puntos de partida y las homotopías utilizadas.

Observación 4.2. Es posible que los métodos de homotopía no sean tan rápidos como sus competidores para este problema. Lo más destacable de este resultado es que da una respuesta positiva a un problema abierto desde hace décadas en álgebra lineal numérica.

References

[1] D. Armentano. Complexity of path-following methods for the eigenvalue problem. *Found. Comput. Math.*, 14(2):185–236, 2014.

[2] D. Armentano, C. Beltrán, P. Bürgisser, F. Cucker, M. Shub A stable, polynomial-time algorithm for the eigenpair problem. aceptado para publicación en *JEMS*.

[3] D. Armentano, F. Cucker. A randomized homotopy for the Hermitian eigenpair problem. *Found. Comput. Math.* 15 (2015), no. 1, 281–312.

[4] S. Batterson. Convergence of the Francis shifted QR algorithm on normal matrices. *Linear Algebra Appl.* 207 (1994), 181–195.

[5] S. Batterson and J. Smillie. The dynamics of Rayleigh quotient iteration. SIAM J. Numer. Anal. 26, 624-636, 1989.

[6] S. Batterson and J. Smillie. Rayleigh quotient iteration for nonsymmetric matrices. *Math.Comp.*, 55(191):169–178, 1990.

[7] C. Beltrán and L.M. Pardo. Smale's 17th problem: average polynomial time to compute affine and projective solutions. *J. Amer. Math. Soc.*, 22(2):363–385, 2009.

[8] C. Beltrán and L.M. Pardo. Fast linear homotopy to find approximate zeros of polynomial systems. *Found. Comput. Math.*, 11(1):95–129, 2011.

[9] L. Blum, F. Cucker, M. Shub, and S. Smale, *Complexity and Real Computation*, Springer-Verlag, New York, 1998.

[10] P. Bürgisser and F. Cucker. On a problem posed by Steve Smale. *Annals of Mathematics*, 174:1785–1836, 2011.

[11] P. Bürgisser and F. Cucker. *Condition*, volume 349 of *Grundlehren der mathematischen Wissenschaften*. Springer-Verlag, Berlin, 2013.

[12] P. Bürgisser, F. Cucker, E. Rocha. On the condition of the zeros of characteristic polynomials. *J. Complexity* 42 (2017), 72–84.

[13] JP. Dedieu. Points fixes, zéros et la méthode de Newton. *Mathématiques & Applications [Mathematics & Applications]*, 54. Springer, Berlin, 2006.

[14] J.W. Demmel. *Applied Numerical Linear Algebra*. SIAM, 1997.

[15] F. Cucker, T. Krick, G. Malajovich, M. Wschebor. A numerical algorithm for zero counting II: Distance to Ill-posedness and Smoothed Analysis, *J. fixed point theory appl.*, 6 (2009)

[16] U. Helmke, J. Moore. Optimization and dynamical systems. (With a foreword by R. Brockett.) *Communications and Control Engineering Series*, Springer-Verlag London, London, 1994. xiv+391 pp.

[17] E. Kostlan. *Complexity theory of numerical linear algebra* J. of Computational and Applied Mathematics 22, 219–230, (1988)

[18] E. Kostlan. *Statistical complexity of dominant eigenvector calculation.* Journal of Complexity 7, 371-379, (1991).

[19] P. Lairez. A deterministic algorithm to compute approximate roots of polynomial systems in polynomial average time. *Found. Comput. Math.* 17 (2017), no. 5, 1265–1292.

[20] A. Ostrowski. On the convergence of Rayleigh quotient iteration for the computa- tion of the characteristic roots and vectors, I. Arch. Rational Mech. Anal. 1, 233-241, (1958)

[21] B. Parlett, W. Kahan. On the convergence of a practical QR algorithm. *Inform. Process. Lett.* 68, 114-118, (1969).

[22] C.W. Pfrang, P. Deift, and G. Menon. *How long does it take to compute the eigenvalues of a random symmetric matrix.* arXiv:1203.4635.

[23] M. Shub. Complexity of Bézout's Theorem VI: geodesics in the condition (number) metric. *Found. Comput. Math.*, 9(2):171–178, 2009.

[24] M. Shub and S. Smale. Complexity of Bézout's Theorem I: geometric aspects. *Journal of the Amer. Math. Soc.*, 6:459–501, 1993.

[25] M. Shub and S. Smale. Complexity of Bézout's Theorem II: volumes and probabilities. In F. Eyssette and A. Galligo, editors, *Computational Algebraic Geometry*, volume 109 of *Progress in Mathematics*, pages 267–285. Birkhäuser, 1993.

[26] M. Shub and S. Smale. Complexity of Bézout's Theorem III: condition number and packing. *Journal of Complexity*, 9:4–14, 1993.
[27] M. Shub and S. Smale. Complexity of Bézout's Theorem IV: probability of success; extensions. *SIAM J. of Numer. Anal.*, 33:128–148, 1996.
[28] M. Shub and S. Smale. Complexity of Bézout's Theorem V: polynomial time. *Theoretical Computer Science*, 133:141–164, 1994.
[29] S. Smale. The fundamental theorem of algebra and complexity theory, *Bull., New Ser., Am. Math. Soc.*, 4(1), 1–36 (1981).
[30] S. Smale. Complexity theory and numerical analysis. In A. Iserles, editor, *Acta Numerica*, pages 523–551. Cambridge University Press, 1997.
[31] S. Smale. Mathematical problems for the next century. *Mathematical Intelligencer*, 20:7–15, 1998.
[32] LN. Trefethen, D. Bau. Numerical linear algebra. *Society for Industrial and Applied Mathematics (SIAM)*, Philadelphia, PA, 1997.

CMat, Facultad de Ciencias, Universidad de la República. Montevideo, Uruguay.
Email address: diego@cmat.edu.uy