ILUIL, *vol. 26, 2003, 919-928*

El Análisis Numérico en los últimos 25 años

ANDREI MARTÍNEZ FINKELSHTEIN
Universidad de Almería

RESUMEN

Este trabajo presenta una breve revisión de los hitos fundamentales del desarrollo del Análisis Numérico en el último cuarto de siglo. Junto con los logros más visibles se recalca el cambio de paradigma que se observa en esta joven ciencia donde los algoritmos finitos son desplazados cada vez más por los métodos iterativos.

ABSTRACT

This paper is a short survey of the main milestones of the development of Numerical Analysis during the last quarter of the century. Besides the most visible achievements, a change of paradigm that can be observed in this young science is stressed, where finite algorithms are being replaced by iterative methods.

Palabras Clave: Matemáticas, Análisis Numérico, Siglo XX.

El Análisis Numérico surge como ciencia... Bueno, aquí tropezamos con el primer obstáculo: ¿qué es el Análisis Numérico? Si es la ciencia del cálculo, entonces sus orígenes coinciden con los propios orígenes del hombre y de las matemáticas. Pero esta definición es demasiado amplia para tener utilidad alguna. Si nos ajustamos a la definición de una autoridad tan indisputable como Trefethen [1992], el Análisis Numérico es el *estudio de algoritmos para los problemas de las matemáticas continuas*. Con esta definición dejamos fuera de consideración aspectos tan computacionales de las matemáticas como la criptografía y teoría de números, los problemas de matemáticas discretas y el álgebra simbólica. Sin embargo, en mi opinión es justo agregar una puntualización a la definición de Trefethen: los algoritmos objeto de estudio del Análisis Numérico están hechos *para ser ejecutados por los ordenadores* o cualesquiera otras máquinas de cálculo. Este matrimonio obligado con los ordenadores supone un paulatino cambio psicológico dentro de la rama. Por ejemplo, desde el colegio conocemos (espero) el método de Gauss para resolver un sistema de

Recibido el 13 de enero de 2003 *I.S.S.N. 0210-8615*

ecuaciones lineales: es fácil de aplicar y fácil de recordar para ejecutarlo de forma manual. No obstante, en la segunda mitad del siglo XX se ha ido imponiendo cada vez más como alternativa la factorización ortogonal (o factorización QR) de la matriz del sistema. El algoritmo utilizado (por ejemplo, el de reflexiones de Householder) es realmente incómodo para ejecutarlo a mano, lo que no es ningún problema para un ordenador, pero posee otras ventajas como la estabilidad o su aplicación para la búsqueda de autovalores.

Por cierto, el método de Gauss es un ejemplo de algoritmo antiguo que se sigue utilizando con éxito en la actualidad. Pero muy cerca tenemos otro algoritmo: el método de Cramer, con el que muchos resolvemos sistemas 2×2 ó 3×3, pero que no aparece en el arsenal de ningún analista numérico que se precie. ¿Por qué? Recordemos que para calcular el determinante de una matriz A $n\times n$ se necesitan $n\times(n!)$ operaciones aritméticas; la regla de Cramer para un sistema $n\times n$ requiere calcular $n+1$ determinantes distintos, con un coste aproximado de $n\times(n+1)!$ operaciones. Si nos enfrentamos a un sistema de 100 ecuaciones con 100 incógnitas (nada grande según los patrones modernos), el algoritmo de Cramer empleará $n\times(n+1)!\approx 9.4259\times 10^{161}$ operaciones. Como referencia, tome nota que un Pentium 4 a 3 GHz realiza unas 5×10^9 operaciones por segundo, de modo que resolver el sistema mencionado por Cramer nos tomaría unos

$$\frac{9.4259\times 10^{161}}{5\times 10^9\times 86400\times 364}=5.9779\times 10^{144}$$

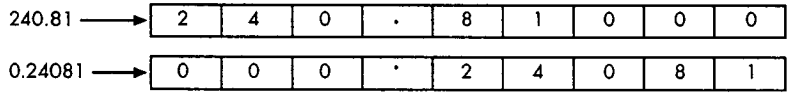
(recuerde que la edad de nuestro sistema solar se estima en $1{,}5\times 10^{10}$ años).

Este ejemplo nos permite además responder a la pregunta natural sobre la necesidad de investigar en el Análisis Numérico en la actualidad, cuando la industria de los ordenadores parece seguir a pie de letra la famosa ley de Moore, duplicando cada 18 meses la potencia de cálculo. Si sólo tuviésemos a nuestra disposición el algoritmo de Cramer, dentro de 15 años tardaríamos en resolver un sistema 100×100 unos 10^{141} años, ¡vaya esperanza! La aparición del algoritmo de Gauss, que necesita tan sólo unas 6.6667×10^5 operaciones (o sea, una fracción de segundo), es un progreso equivalente a más de 500 años de evolución según la ley de Moore. Este ejemplo justifica la máxima que *a medida que las máquinas se vuelven cada vez más potentes, la eficiencia de los algoritmos se hace más y no menos importante.*

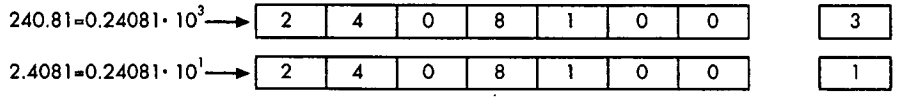
Después de estos ejercicios numéricos, volvamos a lo que nos ocupa: ¿cuáles han sido en realidad los avances más significativos de esta joven ciencia

en los últimos 25 años, es decir, aproximadamente desde 1975? Toda lista de «top 10» tiene una componente subjetiva inevitable dada por las preferencias, conocimientos y ganas de trascender del que la confecciona. Por ello voy a intentar mencionar tan sólo la flor y nata de los últimos logros, aquellos éxitos que han recibido el reconocimiento rápido de la comunidad científica.

Una de las tendencias más notables que ha marcado un cierto grado de madurez del Análisis Numérico ha sido la estandarización a todos los niveles. Es posible que uno de los avances más básicos y más importantes de este período haya sido la proliferación de la **aritmética del punto** (o coma) **flotante**. Un procesador de un ordenador consta de registros de longitud finita donde se almacenan números (típicamente, en base binaria) para operar con ellos. ¿Cómo almacenar valores reales en esos registros? Una opción, tal vez la más natural y la menos eficiente, es repartir el espacio disponible de una vez por todas, asignando tantas casillas (bits) a la parte entera, y el resto a la fraccionaria:



No hace muchos años algunos diseñadores de ordenadores empleaban este método, conocido como aritmética de punto fijo. Mucho más racional es repartir el registro de otra forma, asignando una parte a la información numérica en sí (mantisa), y la otra a la ubicación del punto decimal (exponente). Se trata de la aritmética del punto flotante:



La utilización del punto flotante en los ordenadores se popularizó a mediados de los años 50. En esa época cada fabricante desarrollaba su propia implementación de la aritmética del flotante, con diferentes bases, precisiones y cantidades de bits para la representación, lo que acarreaba serios problemas a la hora de elaborar un software común para los cálculos científicos. Por ejemplo, los modernos PC, los ordenadores DEC o los superordenadores Cray usan la representación binaria, los IBM de la serie 360/370 utilizaban la hexadecimal, y las calculadoras programables de HP, la decimal.

A finales de los 70 y principios de los 80 tuvo lugar una extraordinaria colaboración entre los diseñadores de microprocesadores y los científicos teóricos, coincidiendo con la revolución en la computación producida por la

aparición del PC. Por parte de la industria participaron Apple, DEC, Intel, HP, Motorola y National Semiconductor; los científicos fueron encabezados por William Kanah, de la Universidad de California en Berkeley. El grupo trabajó bajo los auspicios del Instituto de Ingenieros Eléctricos y Electrónicos de los Estados Unidos (en abreviatura, IEEE), y se conocía con las siglas IEEE p754. Ese trabajo concluyó con la publicación en 1985 del estándar #754 del IEEE [1987], mejorado en 1987 con el estándar IEEE 854. Comúnmente se habla simplemente del «estándar IEEE»; utilizado por la mayoría de los PC y estaciones de trabajo modernas, éste fija unas normas para la representación de los números en punto flotante, para la realización de operaciones aritméticas y para el tratamiento de situaciones excepcionales, tales como la división por cero.

Los beneficios de emplear una aritmética bien diseñada y definida con precisión, como en el estándar IEEE, son muy importantes y están bien documentados [Higham, 1996]. En particular, esto permite un mejor manejo de excepciones de la exponente, digamos del desbordamiento («overflow»), similar al que causó la destrucción del cohete Ariane 5 de la Agencia Espacial Europea el 4 de Junio de 1996, tras 40 segundos de vuelo [Demmel]. Además, algoritmos muy importantes pueden perder rapidez o precisión en aritméticas de máquinas que no satisfacen algunos requisitos de dicho estándar, tales como un redondeo correcto de las operaciones de punto flotante. De hecho, recordemos la triste celebridad que obtuvo el flamante microprocesador Pentium de Intel a finales de 1994 cuando por un error en la tabla interna utilizada para implementar las operaciones en punto flotante cometía errores hasta 11 veces mayores de lo estipulado.

A un nivel ligeramente superior otro estándar reciente ha jugado un papel importante: BLAS. La noción de BLAS (Basic Linear Algebra Subprograms) aparece ya en los trabajos de Wilkinson [1948] en los primeros días de los ordenadores digitales, quien sugirió la «preparación de rutinas estándar de generalidad considerable para procesos más importantes que surgen en computación». El primer conjunto estandarizado de BLAS (para Fortran) se publicó en [Lawson, 1979]; éste se conoce ahora como BLAS de nivel 1, e incluye rutinas para operar con vectores (producto escalar, norma, etc.). Los beneficios del uso de este BLAS (sobre todo, modularidad, claridad y aumento de velocidad de cómputo) ya han sido reportados por los creadores de LINPACK. Con posterioridad se han creado BLAS de nivel 2 y 3 con estructura piramidal.

La biblioteca **LAPACK** [ANDERSON *et. al.*, 1995], que apareció en 1992 y que ha sufrido periódicas actualizaciones, fue una colección de rutinas de mayor nivel escritas en Fortran 77 destinadas a la solución de problemas de álgebra lineal. En realidad, esta biblioteca tuvo predecesores en otras similares (LINPACK y EISPACK) de los años 70, mejorando sus funcionalidades, precisión y robustez.

El siguiente hito fundamental en el software destinado al Análisis Numérico ha sido **MATLAB**. Éste comenzó como un paquete interactivo para cálculos matriciales (de ahí su nombre, MATrix LABoratory), desarrollado inicialmente por C. Moler con fines docentes, pero rápidamente cobró notoriedad entre los científicos e ingenieros como una ayuda inestimable en el cálculo científico. MATLAB fue rediseñado a mediados de los años 80 (siendo a partir de ese momento un software comercial), convirtiéndose en un lenguaje de alto nivel para el cálculo científico e incorporando capacidades de visualización y las bibliotecas más actuales tales como LAPACK y ARPACK, así como otras funcionalidades: manejo de matrices dispersas, conexión directa con hardware de control, y el Simulink (módulo para diseño y simulación). Este sistema, que da acceso a algoritmos muy eficientes con pocas líneas de código y permite a los especialistas centrarse directamente en la esencia matemática del problema, sin preocupación por aspectos puramente computacionales tales como declaración de variables o manejo de memoria, se ha convertido en un estándar de facto en la industria. Se afirma que una buena cantidad de cálculos para el diseño de los transbordadores espaciales de la NASA es realizada con MATLAB.

Hoy día somos testigos del nacimiento y desarrollo de la **computación paralela**, aunque por ahora nadie tiene las ideas muy claras sobre cómo programar bien los ordenadores paralelos. Si utilizamos enfoques tradicionales, subdividiendo una tarea en trozos y repartiéndolos entre los procesadores, todo lo que ganamos en la eficiencia de la computación simultánea lo podemos perder en los tiempos de espera, intercambio y recolección de los resultados parciales. No cabe duda que nuestro cerebro (el ordenador más multiprocesador que se puede concebir) no trabaja así, y es de esperar que la Neurobiología y el Análisis Numérico se alimenten unos a otros en este campo de investigación. Mientras, se trabaja en la creación de los fundamentos de la computación paralela: ya existen y se utilizan con éxito los subprogramas básicos del Álgebra Lineal (agrupados en Parallel BLAS y BLACS), las bibliotecas de rutinas ScaLAPACK (subconjunto especializado de LAPACK), etcétera. Varios problemas concretos de Análisis Numérico, tales como la predicción del tiempo,

ya se benefician de la paralelización, y nadie duda que el futuro de la computación de escala está ahí. El gran avance en el proyecto del Genoma Humano, realizado por Celera Genomics, se ha debido tanto a la computación paralela masiva como a algoritmos numéricos muy ingeniosos, permitiéndole adelantar a otros proyectos con subvención estatal. No se puede olvidar el enorme impacto que ha tenido la World Wide Web también en este aspecto de nuestras vidas: investigaciones en criptografía (ruptura de códigos RSA), el proyecto SETI (búsqueda de indicios de civilizaciones extraterrestres) y otros se llevan a cabo hoy día con el «esfuerzo mancomunado» de millones de modestos ordenadores de sobremesa conectados a la red, ejecutando trozos de código en sus momentos de inactividad.

Los ordenadores y los seres humanos tenemos ventajas y diferencias uno frente a otro. Donde los ordenadores nos ganan sin dudas es en la capacidad de repetir una misma operación una y otra vez sin equivocarse. De esta forma, al diseñar los algoritmos modernos los hacemos «cómodos» y fiables para los ordenadores, aunque no necesariamente para nosotros. Sin duda, los ordenadores se sienten mucho más confortables con los métodos iterativos que nosotros. Por ello, una tendencia general que se aprecia en el Análisis Numérico es la siguiente: a *medida que pasa el tiempo, el cálculo científico de gran escala utiliza cada vez más algoritmos aproximados, incluso para problemas que pudiesen ser resueltos en principio con algoritmos finitos y de forma exacta en una cantidad finita de pasos. Los algoritmos aproximados son más robustos que los exactos, y muchas veces, más rápidos.* Es más, aunque exista una fórmula explícita para la solución de un problema dado, no siempre usarla es una buena idea. Durante siglos se buscaron fórmulas (es decir, algoritmos finitos) para calcular las raíces de un polinomio, hasta que Abel truncó las esperanzas de hallar la receta definitiva. Hoy día es raro que alguien se sepa de memoria las pocas fórmulas que se conocen (excepto la del binomio cuadrático), y en la práctica utilizamos métodos iterativos (tales como el de Newton-Raphson), rápidos, robustos y fiables, incluso para las ecuaciones de grado 3 ó 4.

La tendencia antes mencionada, que a primera vista pudiera sonar paradójica, se observa en todas las ramas del cálculo científico.

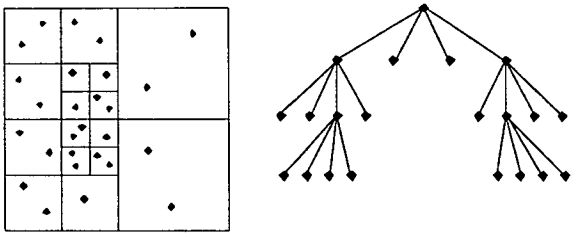
En el Álgebra Lineal Numérica (pilar fundamental de todo el Análisis Numérico) un ejemplo fehaciente lo tenemos en los métodos de los **gradientes conjugados** (y de forma más general, de los gradientes conjugados pre-condicionados o PCG) y de **Lanczos** para la solución de sistemas de ecuaciones lineales

cuya matriz es definida positiva (una subclase muy importante en la práctica). Estos algoritmos descubiertos en los años 50 por Hestenes y Stiefel (PCG) y por Lanczos, fueron concebidos inicialmente como métodos directos, pues garantizan la solución de un sistema $n \times n$ en, a lo sumo, n pasos. Sin embargo, en su implementación práctica esto no se logra por la presencia de los errores de redondeo, de manera que tenemos que concebirlos como métodos iterativos, lo que hizo perder interés inicialmente en dichos procedimientos. Lo curioso es que ninguno de sus creadores predijo en aquel momento la importancia que iban a adquirir estos métodos en un futuro muy próximo. La explicación es sencilla: tanto el método de eliminación de Gauss como el algoritmo de Cholesky necesitan un orden de n^3 operaciones para llegar a la solución, mientras que PCG para ciertas matrices estructuradas puede reducir esta cantidad hasta Cn^2 o incluso menos (eso sí, pudiendo ser C bastante grande). Sus creadores lo sabían, pero en los años 50 los valores de n que se manejaban eran aún demasiado pequeños para que en la práctica $n^3 \gg Cn^2$, pero a medida que los ordenadores se han ido haciendo más y más rápidos y potentes, la apreciación de n «grande» para un analista numérico también ha experimentado evolución, ganando aproximadamente un orden cada 15 años [TREFETHEN and BAU III, 1997]:

Año	n **grande**	**Implementación**
1950	20	Wilkinson
1965	200	Forsythe y Moler
1980	2000	LINPACK
1995	20000	LAPACK
2010	200000	?

En estas condiciones el enorme poder de los métodos PCG y Lanczos no tiene competencia ante matrices muy grandes y dispersas: en muchos casos no es necesario ni llegar a las teóricas n iteraciones, logrando una aproximación a la solución con una gran precisión en bastante menos pasos.

Otro ejemplo lo encontramos en los métodos de multipolos (multipole methods), descubiertos originalmente por Rokhlin y Greengard en los 80 como procedimientos para calcular el movimiento de n cuerpos en interacción gravitacional (o electrostática). Un ataque «frontal» a este problema requiere un orden de n^2 operaciones por paso (cantidad de «parejas» de cuerpos entre los n existentes). La estrategia central del **Fast Multipole Method (FMM)** es la utilización de árboles para agrupar las partículas que interactúan a diferentes escalas espaciales:



Después se calculan las interacciones entre grupos «lejanos» por medio de desarrollos en series, reduciendo la cantidad de operaciones a algo así como *Cn* por iteración. De nuevo, a medida que *n* se hace grande, podemos apreciar (y admirar) cada vez mejor el poder de estos algoritmos.

Los métodos computacionales de optimización no han estado ajenos a esta tendencia. Los problemas de la Programación Lineal son matemáticamente finitos, y desde sus orígenes han sido resueltos por métodos finitos, reinando de forma absoluta e indiscutible el método simplex, desarrollado por Dantzig en los años 40 del siglo XX, coincidiendo con la aparición de los primeros ordenadores digitales. Este algoritmo se basa en el movimiento a lo largo de las aristas del poliedro factible, mejorando en cada desplazamiento, lo que garantiza el encuentro con el mínimo (o máximo) global en una cantidad finita de pasos. Existen numerosas y eficientes implementaciones del método simplex para sistemas grandes, siendo éste uno de los algoritmos más utilizados en la práctica. ¿Quién podía sospechar hace 15 años que iba a aparecer un serio pretendiente al trono? Pero en 1984 Karmarkar [1984] publicó un trabajo, mostrando que los algoritmos iterativos (o sea, infinitos) son en ocasiones mejores que el simplex, dando origen a los llamados hoy día **métodos de punto interior (interior-point methods)** que, como dice su nombre, intentan acercarse al óptimo desde dentro del poliedro factible, y no a lo largo de las aristas de su frontera. De esta forma, los métodos de punto interior para problemas de muchas variables suelen converger en bastante menos iteraciones (con la precisión requerida) que el simplex, aunque cada iteración sea más costosa que en el algoritmo clásico.

Los métodos aproximados penetran y revolucionan todos los aspectos de nuestras vidas. La **transformada rápida de Fourier (FFT)** ha tenido un enorme impacto en la eficiencia de los algoritmos de tratamiento de señales. El descubrimiento y desarrollo de los **wavelets** (bases ortogonales que generalizan a la trigonométrica) ha dado origen al análisis de multiresolución, al tratamiento de

señales variables en el tiempo (tales como el habla), y a la compresión eficiente de información (pensemos en los estándares tan conocidos como MPEG y MP3, que nos permiten disfrutar de las películas en DVD y de toneladas de música almacenada en un solo disco compacto).

El tamaño de este artículo obliga dejar «fuera de borda» otros muchos aspectos y logros del Análisis Numérico moderno, pero tampoco era su objetivo mencionarlos todos. Sólo se pretendía hacer un pequeño balance de cuánto hemos madurado en tan poco tiempo, 25 años. Nuestros medios de cómputo han cambiado de manera extraordinaria: recordemos que a mediados de los gloriosos 70 se veía aun de manera ocasional la regla logarítmica, y volvamos la vista a nuestro ordenador de sobremesa o portátil, por no citar a los titanes que crean películas enteras con actores virtuales o baten a los campeones de ajedrez. Pero todo lo que «saben» hoy estos maravillosos inventos tecnológicos se lo deben a centenares de investigadores, en ocasiones anónimos, que día a día les enseñan a calcular más rápido y mejor, permitiéndonos predecir el tiempo con días de antelación, proyectar la expedición a Marte, usar el móvil (con cámara digital integrada), hacernos un TAC, ver una película en formato digital y muchas cosas más que, no por cotidianas y en ocasiones de utilidad discutible, nos deben impedir admirarlas como conquista de las matemáticas y del saber humano en general.

BIBLIOGRAFÍA

ANDERSON, E.; BAI, Z.; BISCHOF, C.H.; DEMMEL, J.W.; DONGARRA, J.J.; DU CROZ, J.J.; GREENBAUM, A.; HAMMARLING, S.J.; MCKENNEY, A.; OSTROUCHOV, S.; SORENSEN, D.C. (1995) *LAPACK Users' Guide, Release 2.0.* Second edition, SIAM, Philadelphia, PA, USA.

DEMMEL, J.W. «ARIANE 5: Flight 501 Failure», URL: http://www.cs.berkeley.edu/~demmel/ma221/ariane5rep.html

HIGHAM, N. (1996) *Accuracy and Stability of Numerical Algorithms.* SIAM, Philadelphia, PA, USA.

IEEE (1987) *IEEE standard for binary floating-point arithmetic, ANSI/IEEE standard 754-1985.* Technical report, Institute of Electrical and Electronics Engineers, New York, 1985. Reprinted in *SIGPLAN Notices, 22(2),* 9-25.

KARMARKAR, K. (1984) «A new polynomial-time algorithm for linear programming». *Combinatorics, 4,* 373-395.

LAWSON, C.L.; HANSON, R. J.; KINCAID, D. R.; KROGH, F. T. (1979) «Basic Linear Algebra subprograms for Fortran usage». *ACM Trans. Math. Software, 5(3)*, 308-323.

TREFETHEN, L.N. (1992) «The definition of Numerical Análisis». *SIAM News*, Noviembre 1992.

TREFETHEN, L.N.; BAU III, D. (1997) *Numerical Linear Álgebra*. SIAM, Philadelphia, PA, USA.

WILKINSON, J.H. (1948) «The Automatic computing engine at the National Physical Laboratory». *Proc. Roy. Soc. London Ser. A, 195*, 285-285.