ANALISIS EMPIRICO DE ALGORITMOS DE INVERSION DE MATRICES, APLICADOS AL CALCULO DE PROPIEDADES MOLECULARES

M.V. GODOY[1]; P.F. PROVASI[2] y Gustavo A. AUCAR[3]

RESUMEN: En este artículo se evalúa la performance en cuanto a tiempos de ejecución de dos algoritmos de inversión de matrices: uno tradicionalmente utilizado en cálculos de propiedades moleculares y uno nuevo implementado en nuestro grupo de investigación y que utiliza un desarrollo en serie de potencias de los elementos matriciales. Se probaron distintas plataformas, sistemas operativos, compiladores y subrutinas de álgebra lineal (BLAS). Se realizaron cálculos para compuestos modelo que requieren el tratamiento de matrices conteniendo entre 80.000 y 4.000.000 de elementos. Se obtuvieron los tiempos de ejecución de dichos algoritmos a través de la evaluación con el método de propagadores de polarización de propiedades moleculares tipo triplete y singulete. Se observó que el rendimiento del algoritmo de la serie aumenta al crecer la dimensión de la matriz a invertir. Para todas las plataformas utilizadas se obtuvo una performance ganancioso de la serie con respecto al algoritmo tradicional.

ABSTRACT: In this article, the performance of two algorithms for matrix inversion are evaluated considering as a parameter the times of execution: one traditionally used in the calculations of molecular properties and a new one implemented in our research group, that makes use of a series development of matrix elements. Different platforms, operating sistems, compilers and subroutines of lineal algebra (BLAS) were investigated. Calculations for model compounds that require the treatment of matrix containing a number of elements between 80.000 and 4.000.000 was carried out. The calculation of molecular properties of both, singlet and triplet type was used to evaluate the execution time of these algorithms. It was observed that the performance of the series algorithm increases when the dimension of the given matrix grows. For all plataforms, a gainful performance of the series was obtained, compared to the tradicional algorithm.

Palabras claves: Algoritmos, inversión de matrices, propiedades moleculares

Key words: Algorithms, matrix inversion, molecular properties

INTRODUCCIÓN

El cálculo de propiedades moleculares de respuesta lineal dentro de la aproximación Random Phase Approximation RPA con el método de propagadores de polarización posee la siguiente estructura general (Oddershede y Sabin, 1991):

(1) Depto. de Física, Facultad de Ciencias Exactas y Naturales y Agrimensura (UNNE), Av. Libertad 5500, (3400) Corrientes, Argentina. E-mail: mvgodoy@exa.unne.edu.ar

(2) Depto. de Física, Facultad de Ciencias Exactas y Naturales y Agrimensura (UNNE), Av. Libertad 5500, (3400) Corrientes, Argentina. E-mail: patricio@rec.unne.edu.ar

(3) Miembro de la Carrera del Investigador Científico y Tecnológico del CONICET. Depto. de Física, Facultad de Ciencias Exactas y Naturales y Agrimensura (UNNE), Av. Libertad 5500, (3400) Corrientes, Argentina. E-mail: gaa@rec.unne.edu.ar

$$R = \tilde{b}\, P\, b \qquad (1)$$

donde R representa la magnitud física de interés, b el vector que contiene elementos matriciales del operador de perturbación correspondiente a la propiedad estudiada y P la matriz del propagador principal (Oddershede *et al.*, 1984) que puede ser de tipo triplete o singulete. Uno de los cuellos de botella de estos cálculos a nivel *ab initio* se encuentra en la determinación de los elementos del propagador principal. Esto requiere, en la práctica, invertir matrices cuadradas de dimensión muy superior a 10.000 (del orden de 10^8 elementos) para cálculos precisos de la constante de acoplamiento indirecto entre espines nucleares, por ejemplo, aún para compuestos moleculares pequeños. La dimensión de la matriz P varía con N^4 siendo N el número de funciones de base utilizadas. Cálculos propios *ab initio* con el paquete de programas Dalton (Helgaker *et al.*, 1997) indican que para el compuesto CH_4 con 126 funciones de base, la inversión de la matriz del propagador principal consume el 57% del tiempo total de CPU. De hecho que por este motivo se han desarrollado recientemente esquemas alternativos que soslayan esta operación (Jensen *et al.*, 1988). Esto inevitablemente produce la pérdida de la información física contenida en la matriz P.

Si lo que se pretende es sólo reproducir en forma cuantitativa los valores de las propiedades moleculares medidos experimentalmente no resulta importante determinar en forma explícita la matriz P. Sin embargo, cuando se busca determinar si existe (y en que grado) una posible correlación entre las variaciones en los elementos de P y ciertas características de la configuración electrónica molecular, se hace imprescindible calcular la matriz P en forma explícita. Esto es actualmente factible de hacer en las implementaciones del formalismo de propagadores de polarización a nivel semiempírico (Contreras *et al.*, 1993).

La expresión de la matriz P es independiente del tipo de propiedad molecular por estudiar, aunque sí depende de la simetría temporal de los operadores involucrados (Aucar, 1996). Esta matriz puede ser de tipo singulete (para operadores que no dependen del espín electrónico) o triplete (en el caso contrario). En ambos casos sus elementos matriciales contienen expresiones que involucran diferencias de energía de orbitales moleculares e integrales tipo de Coulomb o de Intercambio. En la referencia (Aucar, 1996) se presentó un desarrollo teórico que permitiría calcular la matriz P en forma de serie de potencias. La implementación práctica de dicho formalismo permitió desarrollar un algoritmo optimizado que se detalla en esta comunicación y que mejora la eficiencia en cuanto a tiempos de los cálculos. Este análisis se aplica inicialmente a nivel RPA por ser esta la aproximación más simple y consistente dentro del formalismo de propagadores de polarización (Oddershede *et al.*, 1984). Se compara el rendimiento del cálculo de la matriz P como serie de potencias, con el esquema estandar previamente utilizado en el método semiempírico CLOPPA-X (X = MNDO, AM1, PM3) (Contreras *et al.*, 1993). El análisis de resultados se refiere a cálculos de las contribuciones tanto del término de Contacto de Fermi (propiedad de tipo triplete) como del término Paramagnético Spin Orbital (de tipo singulete) del parámetro espectroscópico J de la Resonancia Magnética Nuclear.

En la sección siguiente se introduce la teoría necesaria sobre desarrollo en serie de potencias aplicable a matrices, luego se presentan las características de los equipos

utilizados para la evaluación de los rendimientos. En la última sección se presenta un análisis de los resultados obtenidos.

Teoría

El aspecto central de este trabajo se refiere al desarrollo en serie de potencias de la matriz P del propagador principal. Se presentan en esta sección los aspectos básicos de la teoría necesaria e implementada en un código computacional para el cálculo de constantes de acoplamiento indirecto entre espines nucleares. Se recurrió a esquemas semiempíricos debido a su menor complejidad en cuanto a la estructura por ser modificada y al hecho de que la metodología desarrollada no pretende lograr resultados cuantitativos en primera instancia sino demostrar su eficiencia en cuanto a los tiempos de ejecución requeridos para obtener la inversa de una matriz de gran tamaño.

Las series de potencias se aplicaron con éxito a números y funciones. La expresión más general para una serie de potencias es:

$$f(x) = a_0 1 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 + \cdots \qquad (2)$$

donde los coeficientes a_1, a_2, a_3,a_4... pueden ser números reales o complejos. Esta serie sólo tiene sentido para aquellos valores de x que hacen que la serie sea convergente a un valor límite. Por este hecho resulta difícil extender este algoritmo a casos más generales; por ejemplo, el caso en que x represente una matriz. Esto último es posible si se tiene en cuenta en forma cuidadosa las restricciones que deben obedecer cada uno de los elementos de la matriz.

A modo de ejemplo podemos considerar una de las series de potencias más simple en la cual todos los coeficientes son iguales:

$$(1-x)^{-1} = 1 + x + x^2 + x^3 + x^4 + \cdots \qquad (3)$$

donde $|x| < 1$

Si en lugar de ser un número x fuera una matriz, digamos X, la nueva condición de convergencia consiste en que los elementos diagonales deben ser los dominantes, es decir, los módulos de los elementos de la diagonal deben ser mayores en términos absolutos que cualquiera de los otros elementos de la matriz. O sea $[X_{ii}] > [X_{ij}]$ y la matriz debe ser no singular (Lowdin, 1964).

Por lo expuesto más arriba cabría el considerar una forma alternativa de invertir una matriz respecto de los métodos tradicionales mucho más costosos, tales como el de Gauss (Press *et al.*, 1992; Burden y Faires, 1996) y sus variantes que incluyen aquellos optimizados para ciertos tipos de matrices. Entre estos se encuentra el método de Cholesky (Press *et al.*, *op. cit.*; Burden y Faires, *op. cit*) para matrices simétricas y definidas positivas. Este último es el que se utiliza en ciertos paquetes de programas como el que implementa el método CLOPPA-X.

En el caso del esquema de inversión de la matriz del propagador principal por serie de potencias, la expresión de la matriz a invertir tiene la forma:

$$\left({}^{m}P_S\right)_{ia,jb} = \left[E^{-1}\left(I - {}^{m}NE^{-1}\right)^{-1}\right]_{ia,jb}$$
$$= \left(E^{-1}\sum_{i=0}^{\infty}\left({}^{m}NE^{-1}\right)^{i}\right)_{ia,jb}$$
$$\approx \left(E^{-1}\sum_{n=0}^{p}\left({}^{m}NE^{-1}\right)^{n}\right)_{ia,jb} = \left({}^{m}P_S\right)_{ia,jb;p} \qquad (4)$$

donde m =1 para propiedades tipo singuletes, m = 3 para propiedades tipo triplete. El índice p se refiere al número de términos de cada serie que corresponde a un dado "camino (ia,jb) de acoplamiento"; E es una matriz diagonal y contiene las diferencias de energía entre orbitales moleculares ocupados y vacantes. ${}^{m}N$ representa la matriz de integrales bielectrónicas.

Una de las condiciones más importantes para obtener un algoritmo más eficiente es considerar el mínimo número de multiplicaciones necesarias para alcanzar la convergencia de la serie utilizada para obtener la inversa. Una estrategia eficiente es la de tener en cuenta las sumas parciales S. La suma parcial de una serie como la de la ecuación (3) se define como:

$$S_n = \sum_{i=1}^{n} x^n$$

Entonces, a partir del primer término definido como I se realizan sumas parciales, tal como se muestra a continuación:

$$S_0 = I$$
$$S_1 = S_0 + x^0 S_0$$
$$S_2 = S_1 + x^1 S_1$$
$$S_3 = S_2 + x^2 S_2$$
$$\ldots\ldots\ldots\ldots$$

Con esta metodología es posible generar la contribución de muchos términos de la serie en muy pocos pasos. Su implementación para el cálculo de la matriz P, junto con consideraciones de optimización es la que se analiza en las secciones siguientes.

Características de hardware y software empleados

Las tiempos de ejecución de los algoritmos han sido calculados trabajando con las distintas plataformas y versiones de subrutinas Blas especificadas en Tabla 1. Se enumeran también los compiladores y las opciones de optimización utilizados a fin de obtener el máximo rendimiento en tiempo de ejecución para un hardware determinado y una configuración de sistema operativo.

Tabla 1: Características del equipamiento utilizado

Plataforma	IBM RISC 6000/370	SGI R10000	
Procesadores	IBM P	MIPS RISC 10000	AMD K6
Ciclos/seg.	66 Mhz	180 Mhz	350 Mhz
Sistema operativo	AIX 3.2	IRIX 6.4	Debian 2.1
Subrutina BLAS	IBM BLAS	SGI BLAS de 64 bits	ATLAS v. 3.0 Beta
Comp. FORTRAN	XL FORTRAN	MISPRO FORTRAN 77	G77 (v. 0.5.24)
Flags	-O3 -qstrict	-O3 –64 –mips4	-funroll-all-loops –O3

El programa que implementa el esquema CLOPPA-X involucra un alto grado de procesamiento de arreglos. Por lo tanto, gran parte de tiempo de la *CPU* es utilizado para el procesamiento de dichos arreglos. Las claves para optimizar este tipo de procesamiento involucran estrategias de minimización del número de ciclos de procesador requeridos para ejecutar los cómputos y del intercambio sucesivo entre capas menos eficientes de la memoria (registros, caché, memoria principal y dispositivos de paginado).

Los compiladores FORTRAN de los distintos sistemas operativos, permiten la utilización de opciones -O,-O2 y -O3 (Tabla 1) que se aplican al lenguaje intermedio producido por este y a nivel de código objeto. Estas proveen un amplio rango de optimizaciones.

Las técnicas que provee el nivel 3 de las subrutinas BLAS permiten la obtención de una performance ganancioso en las operaciones matriz-matriz. El código interno de las mismas es generalmente específico para una arquitectura y por lo tanto están estrechamente ligadas a la performance característica de cada una de ellas.

- IBM BLAS: Son provistas por el fabricante, proveen un adecuado manejo de TLB (Translation Lookkaside Buffer) y bloqueo para diferentes tamaños de cache, en tiempos de ejecución (IBM Canada Ltd. Laboratory, 1993).
- SGI BLAS: Provistas por el fabricante, con direccionamiento a 64 bits proveen técnicas de "blocking" y de "outer-loop" para tomar ventaja de la memoria cache. Para utilizar estas subrutinas el código del programa CLOPPA a sido portado automáticamente en su totalidad a código de 64 bits, de modo de lograr la máxima performance en loops internos, mediante un espacio de direccionamiento virtual más largo y operaciones aritméticas lógicas más veloces.
- ATLAS VERSION 3.0 Beta: Las ATLAS (Automatically Tuned Liner Algebra Software) aplican metodologías altamente eficientes para la construcción de rutinas básicas del álgebra lineal, para las distintas arquitecturas de procesadores, de modo de ordenar la mayor parte de las operaciones en la memoria cache de la maquina sobre la que se han instalado (Whaley y Dongarra,.....).

En la subrutina denominada Dserie se utilizan, para la parte voluminosa del cómputo (las multiplicaciones de matrices), llamadas a la subrutina DGEMM (Double precision Matrix Multiply), que tienen un conjunto estandarizado de parámetros tales como tamaño de matriz y tipo. Las operaciones utilizadas para el producto de matrices tienen la siguiente forma:

$C \leftarrow \alpha AB + \beta C$ donde α y β son escalares, $A(m \times k)$, $B(k \times n)$ y $C(m \times n)$ son matrices (Dongarra *et al.*, 1990).

RESULTADOS Y DISCUSIÓN

El cálculo de la constante de acoplamiento J se puede considerar como dividido en varios procesos. De estos el de mayor consumo de tiempo, es el que realiza la inversión de la matriz de la que resulta P. La relación entre los tiempos de ejecución de esta inversión y el total puede no ser lineal. Esto es lo que se observa en la Tabla 2 donde en las columnas sexta y séptima se observan diferencias importantes, para las matrices de mayor tamaño, que no se traducen en las diferencias entre los tiempos de cálculo totales.(ver Tabla 3, columnas octava y novena).

Tabla 2: Tiempos de ejecución de los algoritmos en las distintas plataformas para propiedades moleculares tipo triplete. Tiempos en segundos.

C[1]	T[2]	IBM RS6000/370			SGI R10000			AMD K6		
		Dsinv	Dserie	R[3]	Dsinv	Dserie	R[3]	Dsinv	Dserie	R[3]
a	289	1,76	4,90	0,38	0,27	1,76	0,15	1,31	6,03	0,22
b	400	5,40	12,91	0,42	0,75	4,41	0,17	4,49	15,49	0,29
c	529	24,18	31,28	0,77	1,94	10,72	0,18	16,03	38,83	0,41
d	676	70,53	63,25	1,11	6,05	18,18	0,33	45,07	63,49	0,71
e	841	163,35	122,72	1,33	16,95	35,29	0,48	111,60	123,61	0,90
f	1024				40,54	67,17	0,60	223,25	220,02	1,01
g	1225				85,89	101,26	0,85	410,02	374,34	1,09
h	1444				172,77	168,66	1,02	705,61	613,00	1,15
i	1681				307,31	258,79	1,19	11155,34	883,53	1,31
j	2209				895,28	594,67	1,50	2756,06	1979,79	1,39

a: 1X,Y-Biciclo [1.1.1] pentano, X = CH_3
b: 1X,Y-Biciclo [1.1.1] pentano X = CH_3
c: 1X,Y-Biciclo [1.2.1] hexano X = CH_3
d: 1X,Y-Biciclo [2.1.2] heptano X =Y = CH_3
e: 1X,4Y-Biciclo [2.2.2] octano X = Y = CH_3
f: 1X,4Y-Biciclo [2.2.2] octano X = CH_3; Y = $CH_2(CH_3)$
g: 1X,4Y-Biciclo [2.2.2] octano X = CH_3; Y = $CH(CH3)_2$
h: 1X,4Y-Biciclo [2.2.2] octano X = CH_3; Y = $C(CH_3)_3$
i: 1X,4Y-Biciclo [2.2.2] octano X = $CH_2(CH_3)$; Y = $C(CH_3)_3$
j: 1X,4Y-Biciclo [2.2.2] octano X = $C(CH_3)$; Y = $C(CH_3)_3$
(1) Compuestos modelo moleculares
(2) Nro. de elementos de cada lado de las matrices
(3) Cociente Dsinv/Dserie

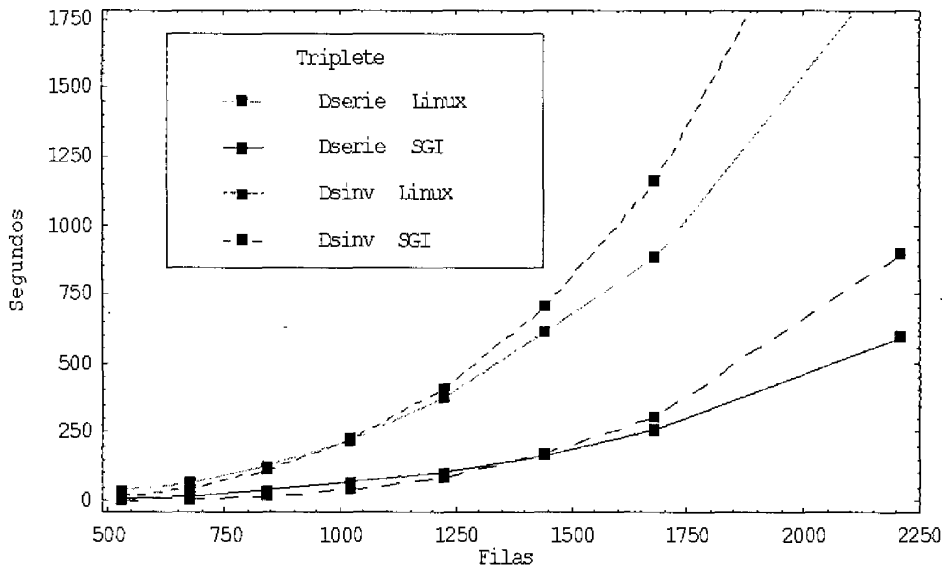

Fig. 1: Curvas de tiempos de ejecución para la inversión de matrices tipo triplete en dos de las plataformas testeadas.

En la Fig. 1 se presentan en ordenadas los tiempos (en segundos) requeridos para calcular la matriz inversa correspondiente a P en forma de serie y con el algoritmo usual encontrado en la literatura (Dsinv), en dos de las plataformas testeadas, en función del número de filas de las matrices a invertir para propiedades moleculares tipo triplete. Resulta evidente que el rendimiento de la Dserie comparado con Dsinv aumenta a medida que crece la dimensión de la matriz a invertir. Como se observa en la Tabla 2, el corte se produce para matrices con un número de elementos superior a $0{,}36 * 10^6$ (IBM RS6000/370), $1*10^6$ elementos (AMD K6) y $2*10^6$ elementos (SGI R10000). En las Tablas 2 y 3 se observa que la diferencia de rendimientos entre el nuevo procedimiento de inversión y el algoritmo implementado con Dsinv no aparece traducido en los tiempos totales de cálculo de J. En particular se logran tiempos relativos de cómputos menores a partir de matrices de dimensión mayor:

- 676 (compuesto d), en la plataforma IBM RS6000
- 1444 (compuesto h) en la plataforma SGI R10000
- 1024 (compuesto f) en el entorno LINUX.

Tabla 3: Tiempos totales de cálculo de J, para propiedades moleculares tipo triplete (en segundos)*

C[1]	IBM RS6000/370			AMD K6			SGI R10000		
	Dsinv	Dserie	R[3]	Dsinv	Dserie	R[3]	Dsinv	Dserie	R[3]
a	37,17	40,64	0,91	14,83	21,00	0,71	8,910	10,44	0,85
b	101,73	109,37	0,93	41,86	54,70	0,76	23,71	27,54	0,86
c	256,87	264,47	0,97	109,46	134,56	0,81	55,80	64,90	0,86
d	567,27	560,78	1,01	243,74	270,00	0,90	120,31	132,45	0,91
e	1133,91	1093,95	1,09	504,60	524,50	0,96	241,29	260,61	0,93
f				883,49	920,45	0,96	458,60	484,49	0,94
g				1583,23	1587,00	0,99	809,80	824,50	0,98
h				2806,95	2791,00	1,00	1352,90	1344,77	1,00
i				4599,01	4568,03	1,01	2208,71	2166,37	1,18
j				10248,00	9825,05	1,04	5302,95	4993,77	1,50

* La nomenclatura es la misma que la utilizada en la Tabla 2.

Esto se debe a los porcentajes del tiempo total que insume el cálculo de la inversa según sea el tamaño de la matriz. Para matrices grandes, como los tiempos de inversión crecen en forma exponencial según se observa en la Fig. 2, este proceso se hace determinante del cálculo en cuanto a su rendimiento final.

En la Fig. 3 se representan las variaciones relativas de crecimiento de las curvas de tiempos de ejecución de los dos algoritmos testeados, tomando como valores bases $n_0 = 83521$ (es el menor tamaño de matriz P considerada); $t_0^1 = 0{,}27$ (tiempo demandado para el cálculo de la inversa de la matriz de tamaño base según Dsinv) y $t_0^2 = 1{,}76$ (idem t_0^1 pero según algoritmo en serie de potencias). Se observa, por ejemplo que al incrementarse n (tamaño de la matriz) en aproximadamente 5 veces y 1/2, la curva Dsinv(n) crece aproximadamente 22 veces y la curva Dserie(n) sólo 10 veces. Se pone de manifiesto que el crecimiento de variación de la curva Dsinv a medida que crece n es siempre significativamente superior a la variación de crecimiento de la curva de tiempos de ejecución según la serie implementada.

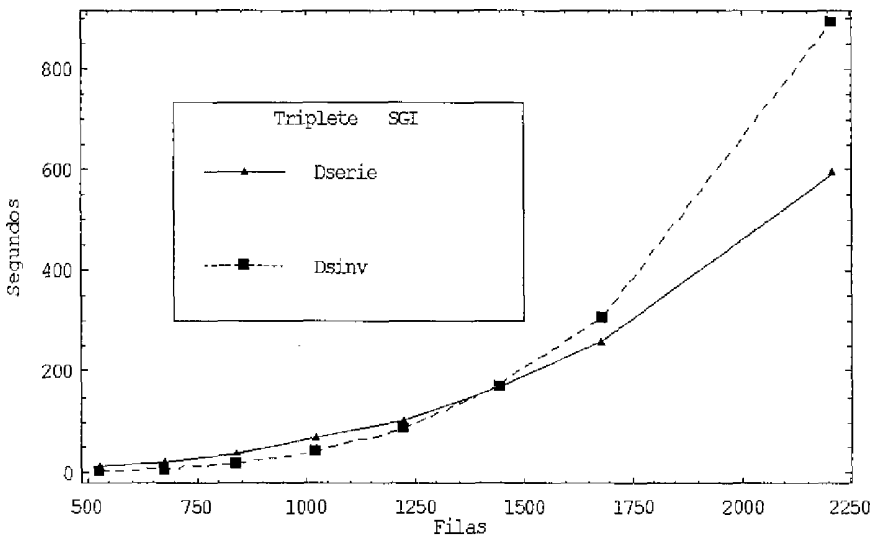

Fig. 2: Curvas de tiempos de ejecución para la inversión de matrices tipo triplete, en la plataforma SGI.

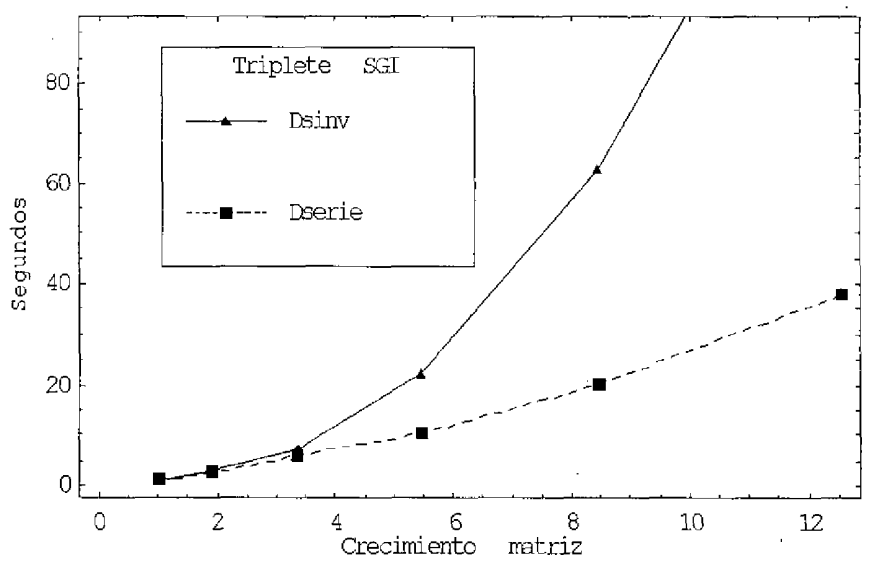

Fig. 3: Curvas de crecimiento relativo de tamaño de matrices vs tiempos de ejecución de calculo de propiedades moleculares tipo triplete y plataforma SGI.

En la Fig. 4 se observan los tiempos de cálculo de las matrices inversas para compuestos del tipo singulete, según los dos algoritmos testeados y en función del número de filas, bajo la plataforma SGI. En este caso el corte de las curvas se produce para compuestos con un menor número de filas que para el tipo triplete, y se explica debido a que es necesario calcular una menor cantidad de términos en la serie para alcanzar la misma cota de error que en el caso de propiedades tipo tripletes. Los valores experimentales obtenidos son mostrados en la Tabla 4.

Allí se puede observar que el porcentaje de tiempo demandado por la Dsinv con respecto al J total aumenta rápidamente al crecer el tamaño de las matrices, sin embargo el de la Serie va disminuyendo aunque más lentamente.

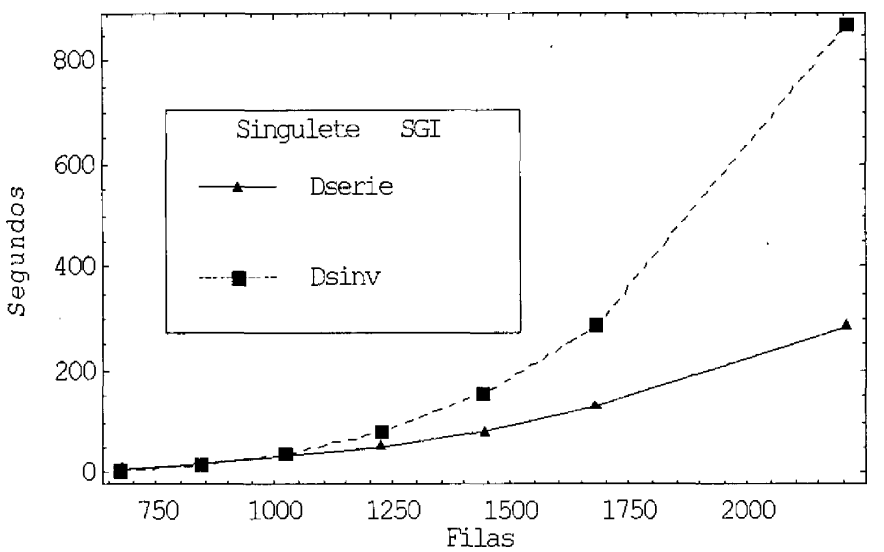

Fig. 4: Curvas de tiempos de ejecución para la inversión de matrices de propiedades moleculares tipo singulete (Plataforma SGI).

Se han realizado cálculos con compuestos cuyas matrices del propagador principal varían entre:

$83521 < n * < 4879681$ elementos y aunque las dimensiones son pequeñas comparadas con el tamaño de las matrices utilizadas en la Química Teórica actual, las tendencias encontradas son bien definidas.

En cuanto a los niveles de significación alcanzados, mediante la generación de 4 términos del desarrollo en serie de la inversa, se han obtenido valores de J con errores relativos que varían entre 1% y 8% para compuestos menores al (g) (dos dígitos enteros de significación); para los compuestos de mayor orden esta diferencia se incremento, hasta llegar solo a un dígito entero de significación.

La relación de la Dserie en la plataforma IBM RS6000/370 y en el entorno LINUX, se mantiene estable en un factor de aproximadamente 3 veces mayor que la Dserie en la plataforma SGI R10000,esto indicaría la importancia de la utilización de objetos generados a 64 bits para determinar el rendimiento de la serie.

Tabla 4: Tiempos de ejecución para propiedades moleculares tipo singulete en plataforma SGI (en segundos)

C(1)	Dserie	J Total	P(2)	Dsinv	J Total	P(3)
a	0,69	10,96	6,29	0,28	10,55	2,65
b	1,81	29,33	6,17	0,76	28,28	2,69
c	4,09	68,18	6,00	1,94	66,03	2,10
d	8,74	143,61	6,08	5,74	140,70	4,08
e	17,10	278,39	6,14	15,45	276,74	5,58
f	31,46	510,03	6,16	37,37	515,94	7,24
g	52,50	874,92	6,00	78,93	901,39	8,75
h	79,98	1392,78	5,74	155,01	1467,81	10,56
i	128,84	2270,62	5,67	285,58	2427,35	11,76
j	285,11	5186,73	5,49	867,81	5769,43	15,04

(1) Compuestos modelo moleculares según se refiere en Tabla 2.
(2) Porcentaje de tiempo demandado para el cálculo de la inversa (según Dserie) con respecto al tiempo J total
(3) Idem anterior (según Dsinv)

Conclusiones

En este trabajo se presenta la implementación de un algoritmo en serie de potencias para el cálculo de inversas de matrices de gran tamaño. Los tiempos de ejecución de esta implementación, testeados para el cálculo de propiedades moleculares con métodos semiempíricos, indican un menor costo computacional y menor complejidad respecto de los algoritmos usados tradicionalmente. A través de las distintas tablas que muestran el tiempo de ejecución del algoritmo, en distintas plataformas, se verifica que la performance de la serie se incremento a medida que aumenta el tamaño de la matriz P. De allí la importancia de su incorporación a métodos de primeros principios.

Se ha logrado optimizar el algoritmo mediante la disminución del número de operaciones elementales de cómputos necesarias en el cálculo de la inversa, al utilizar una estrategia de sumas parciales. Sin embargo, persiste aún una desventaja como es el número de matrices de trabajo utilizadas en el cálculo de la matriz P.

Para cálculos a nivel *ab initio* el comportamiento observado en la subrutina Dsinv (o sus equivalentes) hace a que no se realice propiamente el cálculo de la inversa de la matriz, sino de una manera indirecta. Lo que se ha logrado con el procedimiento de desarrollo en serie podría significar que en un futuro cercano se logre realizar el cálculo explícito de la inversa. De esta manera, se pretenderá luego analizar la física contenida en el propagador principal que ha demostrado ser muy importante para el entendimiento del origen de los parámetros espectroscópicos de RMN.

En nuestro grupo de investigación se continúa trabajando en este sentido y en el tratamiento computacional de métodos *ab initio*. La implementación de este algoritmo dentro de este tipo de métodos permitirá comprender más adecuadamente la física subyacente en la interpretación de importantes propiedades moleculares.

BIBLIOGRAFÍA

AUCAR, G.A.,1996. *Chem. Phys. Letters*, 254:13-20.

BURDEN, R.L. y D.J. FAIRES, 1996. *Análisis Numérico*. Grupo Editorial Iberoamericana, México D.F. 807 p.

CONTRERAS R.H.; G.A. AUCAR; R. LOBAYAN; M. RUIZ de AZUA y C.G. GIRIBET, 1993. *J. Molec. Struc.(Theochem)*, 284:289+.

DONGARRA, J.; J. DU CROZ; I. DUFF y S. HAMMARLING, 1990. A set of level 3 basic linear algebra subprograms. ACM *Trans. Math. Soft., 16* (1): 1-17.

IBM Canada Ltd. Laboratory, 1993. AIX VERSION 3.2 RISC System/6.000. Optimization and Tuning Guide for Fortran, C, and C++.

HELGAKER, T.; H.J.Aa. JENSEN; P. JORGENSEN; J. OLSEN; K. RUUD; H.A. GREN; T. ANDERSEN; K.L. BAK; V. BAKKEN; O. CHRISTIANSEN, P. DAHLE; E.K. DALSKOV; T. ENEVOLDSEN; B. FERNANDEZ; H. HEIBERG; H. HETTEMA; D. JONSSON; S. KIRPEKAR; R. KOBAYASHI; H. KOCH; K.V. MIKKELSEN; P. NORMAN; M.J. PARKER; T. SAUE; P.R. TAYLOR and O. VAHTRAS, 1997. DALTON, an electronic structure program, release 1.0. http://www.kjemi.vio.no/software/dalton/dalton.html

JENSEN, H.J.; H. AGREN y J. OLSEN, 1988. *J. Chem. Phys.*, 89: 36-54.

LOWDIN, P.O., 1964. *J. Molec-Spectroscopy*, 326:13.

ODDERSHEDE, J.; P. JORGENSEN y D.L. YEAGER, 1984. Comput. *Phys. Rep.*, 41:33.

ODDERSHEDE, J. y J.R. SABIN, 1991. *Int. J. Quantum Chem.*, 371: 39.

PRESS, W.H.; B.P. FLANNERY; S.A. TEUKOLSKY y W.T. VETTERLING, 1992. NumeTical *Recipes* in Fortran. Cambridge University Press, Cambridge.

WHALEY, R. y J. DONGARRA. Automatically tuned linear algebra software. http://www.nethb.org/atlas

Recibido/Received/: May-00
Aceptado/Accepted/: Dic-00