Mecánica Computacional Vol 20. pp. 474 — 481
Compilado por Fernando G. Flores
Córdoba, 30 de octubre - 2 de noviembre de 2001

ANÁLISIS DEL COMPORTAMIENTO DE ALGORITMOS PARALELOS DE ELEMENTOS FINITOS CON DESCOMPOSICIÓN DE DOMINIO EN APLICACIONES GEOFÍSICAS

Patricia M. Gauzellino, Juan E. Santos
Departamento de Geofísica Aplicada
Fac. de Cs. Astronómicas y Geofísicas, U.N.L.P.,
Paseo del Bosque s/n, 1900 La Plata, Argentina.
e-mail: gauze@fcaglp.fcaglp.unlp.edu.ar
y
Fabio I. Zyserman
Departamento de Física, U.N.L.P.
C.C. 67, 1900 La Plata, Argentina.
e-mail: zyserman@venus.fisica.unlp.edu.ar

ABSTRACT

The aim of this presentation is to determine the performance of several parallel algorithms, which has been written in Fortran and use the standard MPI, in geophysical applications such as the simulation of the propagation of waves and estimation of electrical conductivities in the subsurface.
The implemented algorithms combine an iterative domain decomposition method with finite elements; the reason to employ parallel strategies is to solve "very large" problems by means of their partition into "smaller" tasks.

RESUMEN

El propósito de esta presentación es determinar la performance de distintos algoritmos paralelos, que escritos en lenguaje Fortran, utilizan el standard MPI para aplicaciones geofísicas, tales como simulación de progación de ondas y cálculo de conductividades eléctricas en el subsuelo terrestre.
Los algoritmos implementados combinan un método iterativo de descomposición de dominio con elementos finitos y el fundamento de emplear estrategias en paralelo se basa en resolver problemas "muy grandes" mediante su fragmentación en un grupo de problemas y tareas "más pequeños".

MARCO TEÓRICO

Estimadores de performance
La propuesta de esta presentación es analizar y comparar, de manera sencilla, algoritmos paralelos que resuelven problemas de gran tamaño y considerable volumen de datos; intentando, además, detectar y explicar posibles defectos de los mismos.
Motivados por su formulación simple, los criterios seleccionados para este análisis de performance son tiempo de ejecución, speed-up, eficiencia y escalabilidad, [1, 2, 3].
Tiempo de ejecución: El tiempo de ejecución de un programa serie es proporcional al tamaño del mismo, $T_s(N)$ y en él habría que diferenciar entre los tiempos de cálculo y de entrada/salida

(input/output) de datos. Esto debe hacerse ya que las sentencias aritméticas son varios órdenes de magnitud más rápidas que las sentencias de input/output (I/O), por lo tanto,

$$T_s(N) = T_{cal}(N) + T_{I/O}(N). \tag{1}$$

El tiempo de ejecución del programa paralelo se mide desde su comienzo en el primer procesador hasta la finalización en todos los procesadores. Entonces, este tiempo es función del tamaño del problema y del número de procesadores, $T_p(N, P)$. En nuestros algoritmos la ejecución se inicia en el procesador 0 encargado de la distribución de los datos de entrada y termina en el mismo procesador con la escritura de los resultados. Así, cada procesador realiza operaciones de cálculo, operaciones de ruteo o comunicaciones y también puede permanecer ocioso. Sumando estos tiempos para todos los procesadores resulta

$$T_p(N, P) = \frac{1}{P} \sum_{i=0}^{P-1} (T^i_{cal} + T^i_{I/O} + T^i_{com} + T^i_{ocio}). \tag{2}$$

T_{cal} depende de las dimensiones del problema y al variar el número de procesadores, de la posibilidad de "mantener" los datos en la memoria cache del procesador, entre otras cosas. El tiempo para transferir mensajes entre procesadores (send/receive), en general, precisa de un tiempo de inicio o latencia de la comunicación y un tiempo de transmisión según la longitud L del mensaje, ya sea en bytes o palabras, es decir,

$$T_{men} = t_s + L\, t_{trans} \tag{3}$$

donde t_s y t_{trans} no son fáciles de obtener, pero se puede afirmar que t_{trans} está en el orden de magnitud de una operación aritmética y t_s es uno a tres órdenes de magnitud más grande. Sin embargo, la latencia pierde significación si los mensajes que se comunican son grandes. Finalmente, el tiempo de ocio puede tener su origen en la falta de cálculos continuos debido a la espera de datos remotos. En los algoritmos propuestos, este tiempo no es importante debido a la carga balanceada de cálculos y comunicaciones.

Speed-Up: Esta medida relaciona el tiempo del programa serie con el del programa paralelo, indicando el factor de reducción al emplear P procesadores. Se define como

$$S(N, P) = \frac{T_s(N)}{T_p(N, P)}. \tag{4}$$

Si $S(N, P) = P$, se tiene speed-up lineal, igualdad que generalmente no ocurre ya que el programa paralelo puede ser más lento que su correspondiente serie (slowdown) debido al exceso de comunicaciones entre procesadores o el programa paralelo es mucho más rápido que su correspondiente serie (speed-up superlineal) debido al uso de la memoria cache en los distintos procesadores.

Cabe citar la conocida ley de Amdahl que establece un valor del speed-up máximo dado por $\frac{1}{S}$, si S da cuenta de la parte secuencial del algoritmo; esto es, si las sentencias que no pueden ser paraleladas corresponden al 10 % del programa, entonces $S(N, P)$ máximo es 10. Con otras palabras, esta ley expresa un hecho que puede intuírse al usar paralelismo, para un problema dado existe un número óptimo de procesadores, aumentar esta cantidad significa desperdiciar el recurso computacional. A partir de estas ideas surgen los conceptos de eficiencia y escalabilidad.

Eficiencia: La eficiencia cuantifica el buen uso se de la computadora paralela y se define de la siguiente manera:

$$E(N, P) = \frac{S(N, P)}{P}, \qquad E(N, P) \leq 1. \tag{5}$$

donde ahora $E(N, P) = 1$ significa máxima eficiencia con todos los procesadores trabajando. Si $E(N, P) < \frac{1}{P}$ se dice que se presenta slowdown.

Escalabilidad: Una forma de obtener este estimador es mediante el análisis del tiempo de ejecución y la eficiencia en función de la cantidad de procesadores para un problema de dimensiones dadas. Generalmente, la eficiencia decrece al aumentar los procesadores y entonces, se puede determinar un adecuado número de ellos. Otra manera de tratar escalabilidad es a partir de la ley de Amdahl, pensando que la única posibilidad de mantener la eficiencia constante es aumentando el tamaño del programa (tal vez implique una reducción de la parte secuencial) al incrementar el número de procesadores. Si se escribe E como

$$E(N,P) = \frac{T_s(N)}{\sum_{i=0}^{P-1}(T_{cal}^i + T_{I/O}^i + T_{com}^i + T_{ocio}^i)}, \tag{6}$$

se observa que su comportamiento constante queda determinado por los "sobretiempos" (overhead) introducidos por el paralelismo (cálculos y réplicas de los mismos , tiempo empleado en comunicar procesadores y tiempo ocioso en ellos).

Diseño de algoritmos paralelos en aplicaciones geofísicas

La utilización de EF para resolver sistemas de ecuaciones diferenciales que surgen del modelado de problemas de interés en geofísica es usual desde hace ya décadas. En este trabajo mostraremos resultados obtenidos en dos aplicaciones diferentes, sísmica y magnetotelúrica, que implican la resolución de las ecuaciones viscoelásticas [4] y las ecuaciones de Maxwell [5] respectivamente. En ambos casos hemos realizado estudios bi y tridimensionales, empleando elementos finitos no conformes (EFNC) [6, 7, 8]. En el caso de magnetotelúrica probaron ser particularmente adecuados para modelar las discontinuidades presentes en la conductividad eléctrica del subsuelo terrestre, ya que, entre otras particularidades, no requieren la continuidad de la componente normal del campo eléctrico, a diferencia de los EF usuales o conformes. En el caso viscoelástico se reduce a la mitad el volumen de datos intercambiados entre procesadores y en el caso 2D se reduce hasta el 25 % el tiempo de CPU.

Los métodos empleados son iterativos [9, 10, 11] y diseñados especialmente para ser utilizados en computadoras paralelas, empleando la técnica de descomposición de dominio (DD), que consiste en dividir el problema global en subproblemas más pequeños, todos de igual tamaño, donde tanto los datos como los cálculos son tratados localmente. De esta manera, en cada elemento de la partición existe concurrencia aunque no independencia, puesto que en general para obtener la solución en un dominio se requieren datos de sus vecinos más próximos.

En nuestras implementaciones la DD más fina posible corresponde a la división del dominio en rectángulos (caso 2D) o paralelepípedos (caso 3D) no solapados. Sobre cada elemento de esta grilla se procesa una cierta cantidad de datos y se realiza un determinado número de operaciones que surgen de las técnicas de elementos finitos empleadas. Como es usual, este proceso desemboca en la resolución de un sistema lineal, cuyo tamaño depende en parte de los EF utilizados. Como hemos mencionado más arriba, la mejor opción probada son los EFNC, donde los grados de libertad se ubican en los centros de los lados de los rectángulos o en los centros de las caras de los paralelepípedos. Así se tienen por elemento, 4 incógnitas para el caso viscoacústico 2D, 8 incógnitas para el caso viscoelástico 2D, 7 incógnitas para el caso viscoacústico 3D, 21 incógnitas para los caso viscoelástico 3D y las ecuaciones de Maxwell 3D, es decir, deben ser resueltos tantos sistemas lineales de los tamaños dados, como elementos hay presentes en la partición. Claramente, asignar un procesador a cada uno de los subdominios no es bueno en estos problemas, ya que casi no tiene trabajo resolviendo un sistema de ecuaciones lineales muy pequeño y el tiempo de comunicación se vuelve dominante, sin tener en cuenta el hecho que se necesita una computadora con cientos de procesadores. Hemos aplicado además diversas técnicas para mejorar el rendimiento de los algoritmos: (i) Relajación, que consiste en cada iteración utilizar en vez de la variable q^{n+1} la expresión

$$q^{n+1} = \alpha q^{n+1} + (1-\alpha)q^n \tag{7}$$

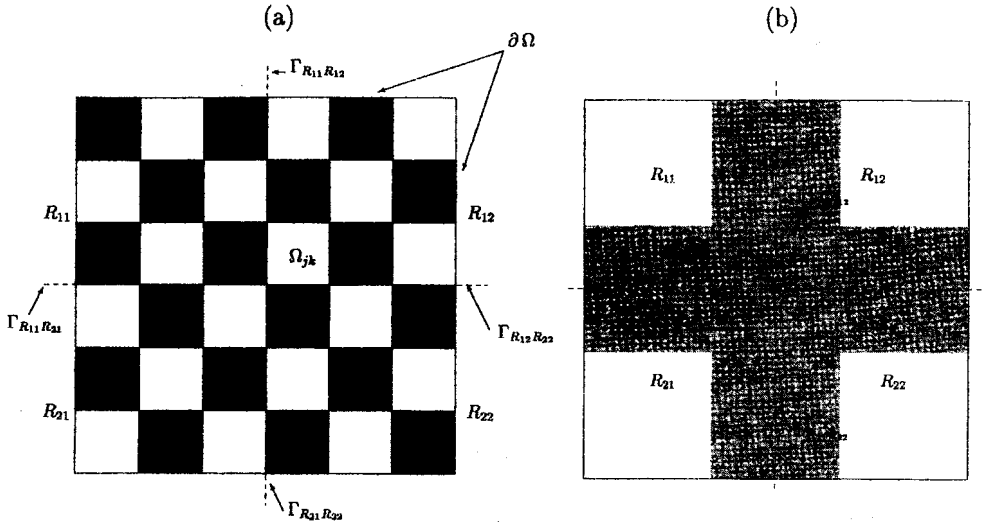

Figura 1: (a) Estructura de la subdivisión del dominio computacional con la técnica *blanco-negro*. En el caso 3D, se está viendo un corte perpendicular al eje mayor de las tiras. Las líneas punteadas representan las "fronteras virtuales" Γ entre las regiones R asignadas a cada procesador. (b) Las celdas sombreadas son las involucradas en el intercambio de información, las flechas muestran su flujo. Observar que sólo una fila y una columna de celdas por región interviene en este proceso

donde α es una constante cuyo valor óptimo se determina experimentalmente. (*ii*) Papas fritas, o tiras: Se agrupan los subdominios en alguna dirección coordenada, esto es, se hace global la partición de elementos finitos y consecuentemente la DD en la dirección coordenada elegida. Esto disminuye el número total de incógnitas y de sistemas a resolver, por lo que la razón "cálculos/comunicaciones" llamada *granularidad* se ve aumentada, lo que influye positivamente en el comportamiento del algoritmo paralelo, ya que una granularidad muy fina implica una caída en el rendimiento por el tiempo excesivo empleado en las comunicaciones. La contracara es el aumento -en gran medida- del tamaño de los sistemas a resolver; sin embargo, siguen siendo mucho menores que en el caso de EF conformes. (*iii*) Técnica *blanco-negro*: los subdominios (o las tiras) se dividen en dos conjuntos, *blanco* y *negro* de modo tal que cada celda blanca queda rodeada de negras. De esta manera, en cada iteración podemos resolver primero para todas las celdas blancas y luego, al calcular las variables en las celdas negras, utilizamos los valores ya actualizados de las celdas vecinas. Por otra parte, la utilización de esta técnica permite hacer uso ventajoso de la memoria cache [12], según se organicen los loops en los algoritmos. Se trata de referenciar elementos próximos del espacio de memoria en intervalos cortos de tiempo. En la Fig. 1 se muestran esquemáticamente lo anteriormente dicho; se puede notar que la DD permite distribuir equitativamente el trabajo entre los procesadores, evitando cálculos y almacenamiento redundantes. Pero como siempre necesita transferir datos de un subdominio a otro, se trata de hacerlo de la manera más eficiente posible, utilizando para ello un patrón de comunicaciones local, estructurado, estático y sincrónico sólo entre un reducido grupo de vecinos.
La estrategia citada conduce a una reducción del tiempo total de ejecución del programa, pues al usar DD se aumenta la localidad sobre los procesadores y se mejora la concurrencia sobre distintos procesadores.

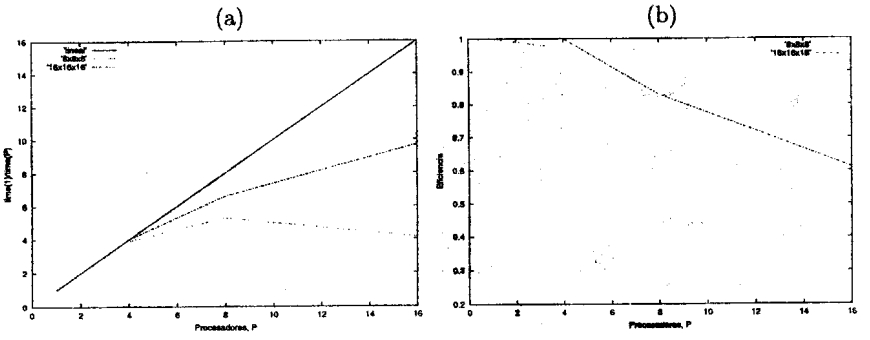

Figura 2: (a) Curvas de speed-up y (b) de eficiencia.

RESULTADOS

En el tratamiento de los problemas de propagación de ondas se tienen en cuenta, entre otros, efectos de atenuación y dispersión; por lo tanto se ha elegido como coordenadas de trabajo el espacio y la frecuencia temporal. Se resuelve para cada frecuencia en particular y dentro de un conjunto cuyo valor central se iguala al de la fuente sísmica usada. De esta manera, al proceso iterativo generado por la DD debe agregarse un bucle en frecuencias, desde los valores más altos a los más bajos.

Asumiendo condiciones de borde absorbentes para ondas que inciden normalmente, se presenta el caso viscoacústico 3D para un modelo geológico que corresponde a una anomalía de baja velocidad en forma de cuña, que difiere de su entorno en $1250m/s$ y donde la ubicación de la fuente y de los receptores en el dominio resulta irrelevante en este análisis. La descripción detallada de estos modelos puede encontrarse en [4] y [11].

En la evaluación del algoritmo (algoritmo I), se resuelve el problema sólo para la frecuencia principal de la fuente, ya que los tiempos de ejecución correspondientes a las restantes frecuencias son similares.

En la figura 2(a) se muestran las curvas de speed-up para el algoritmo I con relajación y tiras B-N, donde se han superpuesto dos tamaños del modelo; 8^3 elementos con 3136 incógnitas y 16^3 elementos con 24832 incógnitas. El speed-up para el programa con el mayor número de incógnitas se aproxima mejor al ideal, pero al aumentar el número de procesadores queda de manifiesto su mal comportamiento. Esto mismo es indicado por el gráfico de eficiencia presentado en 2(b). Entonces, sólo unos muy pocos procesadores (no más de cuatro) resuelven en forma óptima. Al aumentar las dimensiones del problema a 32^3 y 64^3 elementos con 19763 y 1576960 incógnitas, respectivamente; se obtiene speed-up superlineal. La causa de la superlinealidad observada se encuentra en la memoria cache de los procesadores. Al aplicar paralelismo, el problema que resuelve cada procesador disminuye su tamaño, y la velocidad de procesamiento se ve aumentada por la utilización de esta memoria. Por el contrario, el programa serie es demasiado grande para un uso efectivo de este beneficio. La Tabla 1 presenta los tiempos de ejecución obtenidos al variar el números de procesadores. Para analizar el tiempo empleado en las comunicaciones interprocesadores se debe contabilizar la cantidad de veces que se transfieren mensajes por el tiempo de transmisión de los mismos y si las hubiera, operaciones globales; tales como la sumación paralela que se lleva a cabo para estimar el error de cada iteración. La ventaja de no realizar particiones en una dimensión determinada (en nuestro caso x), se pone de manifiesto al

Procesadores	32×32×32	64×64×64
1	122.5	2062.2
4	14.0	284.6
8	7.1	187.1
16	6.4	82.9

Tabla 1: Tiempos de ejecución en segundos para el modelo viscoacústico 3D.

Algoritmo EFNC masivo	T_{com}
Granularidad más fina	$11520(t_s + 2\ t_{trans})$
De a 2 en x, y y z	$5376(t_s + 8\ t_{trans})$
De a 4 en x, y y z	$2304(t_s + 32\ t_{trans})$
De a 8 en x, y y z	$768(t_s + 128\ t_{trans})$
Algoritmo EFNC tiras	T_{com}
Granularidad más fina	$480(t_s + 32\ t_{trans})$
De a 2 en y y z	$224(t_s + 64\ t_{trans})$
De a 4 en y y z	$96(t_s + 128\ t_{trans})$
De a 8 en y y z	$32(t_s + 256\ t_{trans})$

Tabla 2: Tiempos teóricos de comunicación entre procesadores para diferentes algoritmos EFNC.

comparar estos tiempos para los algoritmos I EFNC masivo y EFNC tiras que se ilustran en la Tabla 2 para distintas agrupaciones entre los 16×16×16 elementos de un dominio. Los factor que preceden al t_{trans} representan a los volúmenes de datos transmitidos, comenzando con dos variables complejas en el caso de la descomposición más fina posible. Además, se puede observar como aumenta el tiempo de las comunicaciones al incrementar el número de procesadores. Lo expuesto parece indicar que habrá una cierta cantidad máxima de procesadores que minimiza los costos. La escalabilidad se muestra en la Tabla 3 donde para diferentes grillas y procesadores se indica el porcentaje de su parte secuencial y su correspondiente speed-up máximo.

Para los resultados del algoritmo II que se muestran a continuación se utilizan dos modelos diferentes: uno en el que existe una anomalía conductora de tamaño relativamente pequeño inmersa en un sustrato homogéneo (Modelo 1), y otro en donde hay presentes dos anomalías de gran tamaño, una conductora y otra resistiva respecto de la capa en la que están inmersos, y debajo de ésta existen una secuencia de sustratos con distintas conductividades (Modelo 2). Para detalles de los mismos, ver [5]. En la Tabla 4 se muestra el comportamiento de las distintas variantes del algoritmo, donde se han utilizado diferentes técnicas de aceleración de la convergencia. En este caso se usa el Modelo 1 y la malla considerada es homogénea, de 64^3

Procesadores	Grilla	Parte sec. (%)	Speed-up max.
4	16 × 16 × 16	0.01	100
8	16 × 16 × 16	0.015	66.7
16	16 × 16 × 16	0.03	33.3
4	32 × 32 × 32	0.003	333
8	32 × 32 × 32	0.004	250
4	64 × 64 × 48	0.0019	526
8	64 × 64 × 48	0.0020	500

Tabla 3: Parte secuencial y speed-up máximo para distintas grillas y procesadores.

Algoritmo II	Iteraciones
DDFE masivo	2640
DDFE masivo con relajación	223
DDFE tiras	286
DDFE tiras con relajación y B-N	130

Tabla 4: Número de iteraciones necesarias para la convergencia en las distintas implementaciones del algoritmo DDFE.

Procesadores	Modelo 1	Modelo 2
1	520.6	2008.5
4	158.3	617.08
8	75.39	312.31
16	39.04	149.84
28	—	85.58
32	20.34	—
56	—	47.44
64	11.65	—

Tabla 5: Tiempos de cálculo en segundos para los modelos descriptos.

elementos. Se observa claramente la mejora en los resultados cuando se utilizan las papas fritas combinadas con relajación, y *blanco-negro*. En la Tabla 5 se detalla el comportamiento de la implementación más eficiente del algoritmo paralelo para los modelos mencionados. Se ha dicho más arriba, sin embargo, que de un algoritmo paralelo debe también esperarse que sea escalable, es decir, que no disminuya su eficiencia a medida que aumentan simultáneamente el tamaño del problema y el número de procesadores. Para analizar el algoritmo II- DDFE en su versión más eficiente- hemos resuelto el Modelo 1 con 8 procesadores en una grilla de 64 $\times$ 64 $\times$ 48 elementos, lo que equivale a unas $3.5 \cdot 10^6$ incógnitas, obteniendo los resultados que se detallan en la Tabla 6.

CONCLUSIONES

Se ha estudiado la performance de distintos algoritmos paralelos -que utilizan el standard MPI- para aplicaciones geofísicas, en estudios de progación de ondas y cálculo de conductividades eléctricas en el subsuelo terrestre.
Los algoritmos implementados combinan un método iterativo con descomposición de dominio y elementos finitos, teniéndose que resolver sistemas lineales muy pequeños.
Los ejemplos numéricos presentados muestran que la técnica empleada, junto con las diversas estrategias para acelerar la convergencia (relajación, papas-fritas, blanco-negro), resultan en algoritmos de muy buen rendimiento en máquinas paralelas. Resulta muy interesante, en el caso electromagnético, el comportamiento escalable del algoritmo, lo que lo habilita a resolver

Procesadores	Grilla	Incógnitas	Eficiencia
8	64 $\times$ 64 $\times$ 48	$3.5 \cdot 10^6$	0.85
16	180 $\times$ 80 $\times$ 60	$6.9 \cdot 10^6$	0.87

Tabla 6: Eficiencia del algoritmo II.

problemas "grandes" sin pérdida de eficienca.

AGRADECIMIENTOS

Los programas paralelos fueron corridos en la SP/2 de la Universidad de Purdue, EEUU. JS y FZ agradecen soporte del CONICET, y FZ también del Depto. de Matemáticas de la UNLP.

REFERENCIAS

[1] **Foster, I.**, *Designing and building parallel programs*, Addison-Wesley, 1995.

[2] **Tinetti, F. y De Giusti, A.**, *Procesamiento paralelo*, Ed. Exactas, UNLP, 1998.

[3] **Pacheco, P.**, *Parallel programming with MPI*, Morgan Kaufmann Publishers Inc., 1999.

[4] **Gauzellino, P.**, *Simulación Numérica de Fenómenos de Propagación de Ondas en Medios Dispersivos*, Tesis doctoral, UNLP, 1999.

[5] **Zyserman, F.**, *Modelado numérico de difusión electrmagnética en el subsuelo terrestre*, Tesis doctoral, UNLP, 2000.

[6] **Santos, J.**, *Global and domain decomposed mixed methods for the solution of Maxwell's equations with application to magnetotellurics*, Numerical Methods for Partial Differential Equations, **14**, 1998, 407-437.

[7] **Douglas, Jr., Santos, J. y Sheen, D.**, *A nonconforming mixed finite element method for Maxwell's equations*, Math. Mod. and Meth. in App. Sci. **10**, 2000, 593-613.

[8] **Douglas, J. Jr., Santos, J. y Sheen, D.**, *Nonconforming Galerkin Methods for the Helmholtz equation*, Numerical Methods for Partial Differential Equations, 2001. (en prensa)

[9] **Zyserman, F., Guarracino, L. y Santos, J.**, *A hybridized mixed finite element domain decomposed method for two dimensional magnetotelluric modelling*, Earth, Planets and Space, **51** 4, 1999, 297-306.

[10] **Zyserman, F. y Santos, J.**, *Parallel finite element algorithm with domain decomposition for three dimensional magnetotelluric modelling*, Journal of Applied Geophysics **44** 4, 2000, 337-351.

[11] **Gauzellino, P. y Santos, J.**, *Frequency domain wave propagation modelling in exploration seismology*, Journal of Computational Acoustics, 2001. (en prensa)

[12] **Douglas, C.**, *Caching in with multigrid algorithms: problems in two dimensions*, Comunicación privada.