Asociación Argentina
de Mecánica Computacional

Mecánica Computacional Vol XXXVII, págs. 469-478 (artículo completo)
A. Cardona, L. Garelli, J.M. Gimenez, P.A. Kler, S. Márquez Damián, M.A. Storti (Eds.)
Santa Fe, 5-7 Noviembre 2019

ACELERACIÓN DE ALGORITMOS DE RESOLUCIÓN DE SISTEMAS LINEALES DE MATRICES DISPERSAS PARA CÓDIGOS DE COMBUSTIBLES NUCLEARES

ACCELARATION OF SOLVER ALGORITHMS OF LINEAR SYSTEMS OF SPARSE MATRICES FOR NUCLEAR FUEL CODES

Matías E. Loza Peralta[a], Martín Lemes Lapasta[a] y Alejandro Soba[a]

[a] *Sección Códigos y Modelos, Gerencia Ciclo del Combustible Nuclear, CAC, CNEA, Av. Gral. Paz 1499, 1650 Buenos Aires, Argentina, loza@cnea.gov.ar, lemes@cnea.gov.ar, soba@cnea.gov.ar*

Palabras clave: sistemas lineales sparse, optimización, combustibles nucleares, DIONISIO.

Resumen. La resolución de ecuaciones diferenciales por métodos discretos como el de elementos finitos requiere la resolución de grandes sistemas de ecuaciones lineales en general dispersas. Tal es el caso de las obtenidas en el desarrollo de códigos de combustibles nucleares, especialmente cuando se trabaja en tres dimensiones. Para acelerar el cálculo minimizando el tiempo de resolución de dichos sistemas es necesario programar *solvers* de ecuaciones lineales eficientes, basados en métodos iterativos entre los que destacan aquellos que utilizan espacios de Krylov, ya que satisfacen las condiciones de velocidad y eficiencia requeridas. Así mismo y para optimizar aún más estos algoritmos, es posible incluir herramientas de paralelización, sea mediante métodos que utilizan memoria distribuida (MPI) o memoria compartida (Openmp). En este trabajo se presentan los resultados obtenidos durante el desarrollo de varios *solvers* iterativos, el CG (gradientes conjugados), el BiCG (gradientes biconjugados) y el BiCGSTAB (gradientes biconjugados estabilizado). Los tres *solvers* se utilizaron para resolver problemas originados dentro del Código de combustibles nucleares DIONISIO, usando distintos tipos de precondicionadores y aplicados a sistemas lineales simétricos y asimétricos, comparándose tiempo y eficiencia de los mismos.

Keywords: sparse lineal systems, optimization, nuclear fuel, DIONISIO.

Abstract. Resolution of differential equations for discrete methods such as finite element one requires the resolution of large linear equation systems in general sparse. This is the case of those ones obtained in the development of nuclear fuel codes, especially at three dimensional works. For accelerate the calculus minimizing resolution time of mentioned systems, it is necessary to program efficient linear equation *solvers*, based on iterative methods among those that stand out those that use Krylov spaces, since they satisfy required velocity and efficiency conditions. Likewise and for further optimize those algorithms, it is possible to include parallelization tools, either through methods that use distributed memory (MPI) or shared memory (OPENMP). In this work, results obtained during the development of several iterative methods, CG (conjugated gradients), BiCG (biconjugated gradients) and BiCGSTAB (stabilized biconjugated gradients), are presented. The three *solvers* were used to solve problems originated inside nuclear fuel code DIONISIO, using different types of preconditioners and applied to symmetric and asymmetric lineal systems, comparing their time and efficiency.

Copyright © 2019 Asociación Argentina de Mecánica Computacional http://www.amcaonline.org.ar

1 INTRODUCCIÓN

La resolución de grandes sistemas de ecuaciones lineales esparcidas, como las generadas por el método de elementos finitos, especialmente en problemas tridimensionales, requiere de algoritmos eficientes y crecientes capacidades de cálculo. La paralelización mediante métodos distribuidos o mediante esquemas de memoria compartida provee una forma eficaz de aprovechar los modernos procesadores *multithreads* o los superordenadores cada vez más accesibles. Concomitantemente, en especial cuando tratamos de resolver sistemas característicos en problemas determinados, resulta conveniente invertir tiempo en mejorar los algoritmos utilizados para ello o hallar precondicionadores que permitan disminuir el número de iteraciones necesarias para alcanzar convergencia. En este sentido se enmarca este trabajo, que revisa en profundidad los *solvers* empleados dentro del código DIONISIO (Soba, Lemes, González, & Denis, 2016) y prueba la eficiencia de nuevos precondicionadores en el tipo de sistemas que el código resuelve. DIONISIO es un código de combustibles íntegramente desarrollado por la Sección Códigos y Modelos de la Gerencia de Ciclo de Combustible Nuclear e incluye modelos termomecánicos y termoquímicos relacionados a los distintos fenómenos que suceden en un combustible nuclear bajo irradiación, muchos de los cuales se resuelven mediante elementos finitos en dominios bi y tridimensionales, estos últimos los más demandantes de poder de computo (Goldberg, Loza Peralta, & Soba, 2019).

2 DESARROLLO

Originalmente el *solver* principal utilizado en el código DIONISIO está basado en la familia denominada Gradiente Conjugado y biconjugado (Magoulès, Roux, & Houzeaux, 2014) con precondicionador de Jacobi [BiCG3]. Estos *solvers* garantizan una eficiencia adecuada en sistemas de matrices ralas, diagonal dominantes, simétricas y no simétricas, aunque al aumentar la cantidad de no nulos en el sistema, el tiempo de resolución y la cantidad de iteraciones aumentan. Por esa razón y en el marco de este estudio, se han incluido nuevos precondicionadores basados en el método de LU incompleto, ILU(0) [BiCGLU2] (Saad, 2003), y nuevos métodos como el del gradiente biconjugado estabilizado [BiCGSTAB](Magoulès et al., 2014) y el método de cuasi mínimo residuo [QMR](García León, 2003), estos dos últimos con precondicionador de Jacobi.

Cabe destacar que trabajar con el precondicionador ILU(0) requiere más requerimientos de cálculo y de memoria, por lo que a igual cantidad de iteraciones que un algoritmo precondicionado con Jacobi, presenta mayores tiempos de ejecución. Además, no se utilizaron algoritmos de reordenamiento en ninguno de los algoritmos. Esto podría influir especialmente en la convergencia del algoritmo BiCGLU2.

2.1 Algoritmos seriales

Se ejecutaron los algoritmos de resolución para distintas matrices obtenidas de problemas de transferencia térmica por conducción de barras combustibles. Este tipo de problemas genera matrices simétricas (Reddy, 1993). Para resolverlos se utilizan distintos mallados surgidos de una barra completa por un lado, y de la zona superior (identificada como *plenum*) de una barra, por otro, integrada por las últimas dos pastillas, el *plenum*, la vaina que recubre a los anteriores y el tapón superior. Los mallados se realizaban con tetraedros y poseen 5.637 y 285.265 nodos para el caso del *plenum*, y 81.211 y 405.337 nodos en el caso de la barra completa. En la Figura 1 se pueden observar la dispersión de los distintos elementos no nulos de cada matriz.

De esos resultados se tomaron la cantidad de iteraciones y el tiempo de ejecución de cada

Copyright © 2019 Asociación Argentina de Mecánica Computacional http://www.amcaonline.org.ar

método para cada matriz considerando un error relativo de 10^{-10} para la norma del residuo. Además, se realizaron otras ejecuciones con una tolerancia de 10^{-14} y se graficó la evolución del error de cada método a lo largo de las iteraciones para cada matriz.

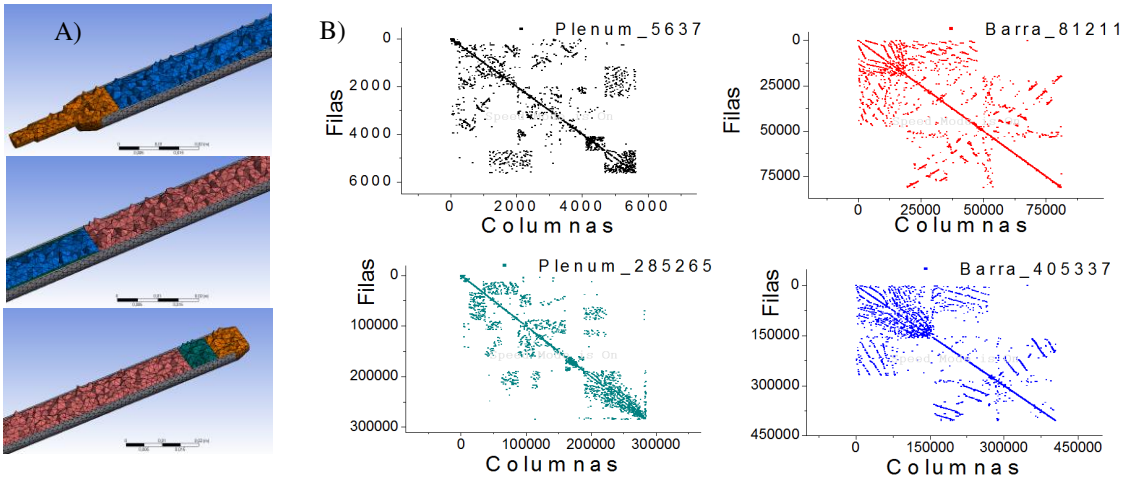

Figura 1: A) Tres imágenes del mallado de tetraedros. B) Cuatro gráficos de dispersión de elementos no nulos de distintas matrices dispersas surgidas de distintos mallados

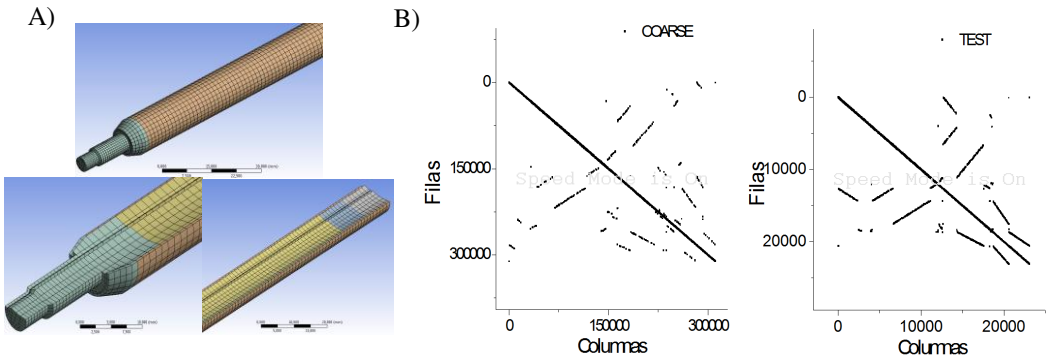

Figura 2: A) Tres imágenes del mallado de hexaedros. B) Dos gráficos de dispersión de elementos no nulos de las matrices dispersas surgidas de DIONISIO

En una segunda instancia, después de testear los algoritmos externos, se incluyeron los *solvers* en el código DIONISIO para resolver algunos de los problemas tridimensionales que el código encara, por ejemplo el de transferencia térmica en el *plenum*. La dispersión de las matrices que surgen de los distintos mallados de la zona del *plenum* se muestra en la Figura 2. Dichos mallados se denominan COARSE y TEST. Se promedió la cantidad de iteraciones y el tiempo de ejecución para todas las iteraciones que realiza el código de la resolución del problema en una corrida del programa. Cabe destacar que la tolerancia de los errores del código es de 10^{-8}.

Por otro lado, se extrajeron las matrices resueltas por DIONISIO y se resolvieron a parte del código para analizar la convergencia de dichas matrices.

En la Tabla 1 se presenta un análisis de las matrices simétricas estudiadas, destacándose la dimensión, la cantidad de no nulos y el factor de no nulos.

Diversos problemas dentro del código DIONISIO generan matrices asimétricas, por lo que

Copyright © 2019 Asociación Argentina de Mecánica Computacional http://www.amcaonline.org.ar

se procedió testear los algoritmos propuestos con este tipo de matrices. Se realizaron ejecuciones de los algoritmos de distintas matrices asimétricas obtenidas de una página web especializada («Matrix Market», 2019) para analizar la convergencia. Entre aquellas matrices destacamos HOR 131, ORSIRR 2, SHERMAN4 y STEAM3. La dispersión de los elementos no nulos de las mismas se pueden observar en la Figura 3. No se analizan los tiempos de ejecución dado que las matrices son relativamente pequeñas.

	Total	Nodos	No Nulos	Factor No nulos
Plenum_5637	75.317	5.637	69.680	2,19E-03
Barra_81211	1.069.085	81.211	987.874	1,50E-04
Plenum_285265	4.091.045	285.265	3.805.780	4,68E-05
Barra_405337	5.543.299	405.337	5.137.962	3,13E-05
COARSE	8.095.420	311.535	7.783.885	8,02E-05
TEST	577.312	23.146	554.166	1,03E-03

Tabla 1: Análisis de las matrices dispersas simétricas evaluadas

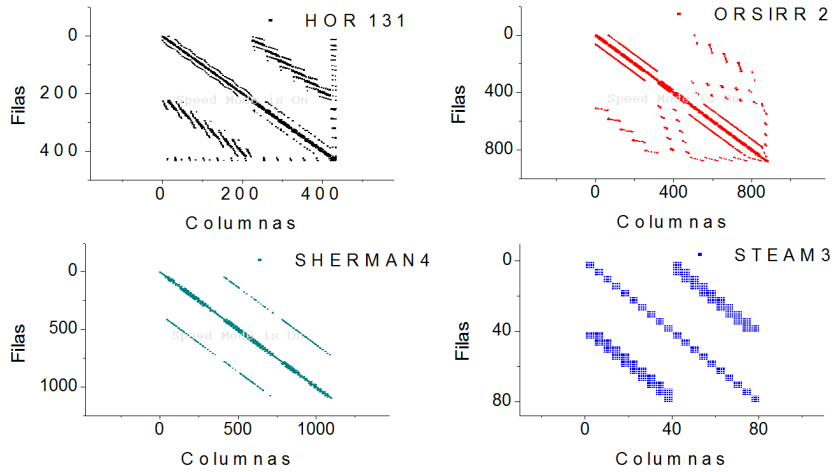

Figura 3: Dispersión de elementos no nulos de las matrices HOR 131, ORSIRR 2, SHERMAN4 y STEAM3

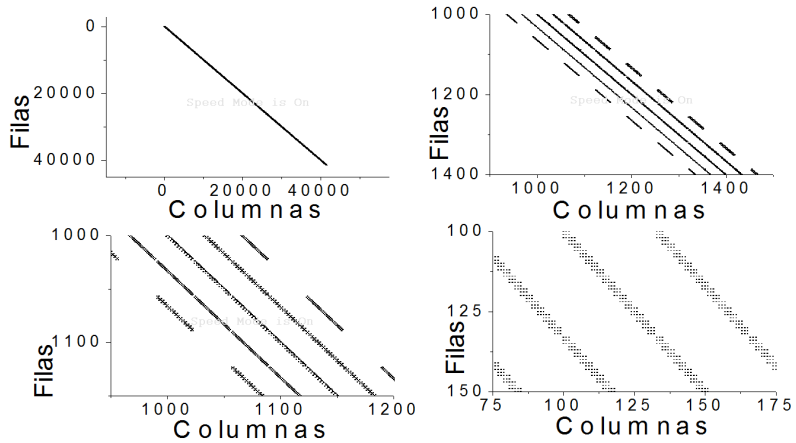

Figura 4: Cuatro gráficos de dispersión de elementos no nulos de la misma matriz con diferente zoom

Copyright © 2019 Asociación Argentina de Mecánica Computacional http://www.amcaonline.org.ar

Se ejecutaron los algoritmos seriales de resolución para distintas matrices obtenidas de problemas de convección-difusión de una barra combustible representada por una malla axisimétrica con elementos rectangulares lagrangeanos de 9 nodos. Estos problemas generan matrices asimétricas (Heinrich, 1980) y particularmente, el problema analizado presenta matrices donde las diagonales relativamente próximas a la diagonal principal son densas como se puede observar en un ejemplo con 41.547 nodos presentado por la Figura 4. Se confeccionaron 3 matrices denominadas CD-41547, CD-15147 y CD-8547 donde el número representa la cantidad de nodos de la malla.

En la Tabla 2 se presenta un análisis de las matrices asimétricas estudiadas similar al de la Tabla 1.

	Total	Nodos	No Nulos	Factor No nulos
HOR 131	4.710	434	4.276	2,27E-02
ORSIRR 2	5.970	886	5.084	6,48E-03
SHERMAN4	3.786	1.104	2.682	2,20E-03
STEAM3	928	80	848	1,33E-01
CD-41547	649.257	41.547	607.710	3,52E-04
CD-15147	236.457	15.147	221.310	9,65E-04
CD-8547	133.257	8.547	124.710	1,71E-03

Tabla 2: Análisis de las matrices dispersas asimétricas evaluadas

2.2 Algoritmos paralelizados

Se paralelizaron los algoritmos de BiCG3, BiCGSTAB y QMR mediante instrucciones de memoria compartida (Openmp (Lewis, 2019)). Se modificó además en los casos de BiCG3 y QMR el tratamiento de la matriz traspuesta que lleva el algoritmo (Magoulès et al., 2014) para poder mejorar la paralelización. Se ejecutaron los tres algoritmos en forma serial y en paralelo de 2 a 8 *theards* para la matriz COARSE y se analizaron los tiempos de ejecución.

3 RESULTADOS DE LOS ALGORITMOS SERIALES

3.1 Problemas de conducción térmica

El análisis se desarrolló tomando cada uno de los *solvers* desarrollados aplicados a resolver las mismas matrices. De cada sistema se analizaron los datos de tiempo y la cantidad de iteraciones de las matrices presentadas en la Figura 1. Algunos de esos datos se presentan en la Tabla 3. En ella se puede observar que el BiCGSTAB presenta los menores tiempos de ejecución en tres de las cuatro matrices analizadas. Por otro lado el algoritmo de QMR es el que presentó tiempos mayores de ejecución.

	BiCG3		BiCGLU2		BiCGSTAB		QMR	
	it	tiempo	it	tiempo	it	Tiempo	It	tiempo
Plenum_5637	48	0,0624004	16	0,0936006	34	0,0468003	47	0,1092007
Barra_81211	336	8,0340515	135	8,6424554	369	8,7048558	340	11,6376746
Plenum_285265	178	17,7529138	59	16,6609068	108	12,1836781	173	25,2253617
Barra_405337	72	10,8264694	23	10,9824704	47	7,176046	72	16,0369028

Tabla 3: Tiempo de ejecución (seg) y cantidad de iteraciones de los algoritmos de resolución para distintas matrices

Copyright © 2019 Asociación Argentina de Mecánica Computacional http://www.amcaonline.org.ar

En la Figura 5 se observa que los gráficos de convergencia agrupados por las matrices estudiadas. En todos se evidencia que el BiCGLU2 presenta menor cantidad de iteraciones. Esto se debe a que el precondicionador es más parecido a la matriz original. Sin embargo, al guardar más valores, también realiza más cálculos lo que hace que los tiempos de ejecución sean mayores como se observa en la Tabla 3. Otro aspecto destacado es la superposición de prácticamente todas las curvas de convergencia de BiCG3 y QMR. Si bien la superposición no es perfecta se observa que los errores evolucionan de manera similar. En las tres matrices donde el BiCGSTAB presentó menores tiempos de ejecución también presentó menor cantidad de iteraciones.

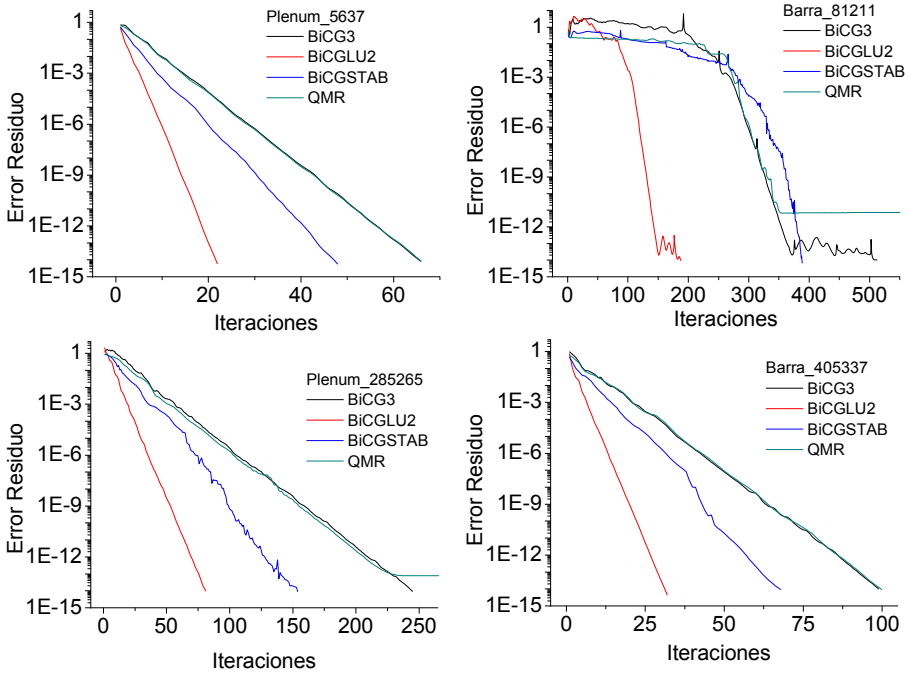

Figura 5: Convergencia de los distintos algoritmos de resolución agrupados por matriz resuelta. *Solvers*: BiCG3 (en negro), BiCGLU2 (en rojo), BiCGSTAB (en azul) y QMR (en verde). Matrices utilizadas: mallado con tetraedros de *plenum* de 5.637 nodos (Plenum_5637, gráfico superior izquierdo) y 285.265 nodos (Plenum_286265, gráfico inferior izquierdo), y de una barra completa de 81.211 nodos (Barra_81211, gráfico superior derecho) y 405.337 nodos (Barra_405337, gráfico inferior derecho)

En la Figura 6 se presentan los gráficos de convergencia agrupados por método de resolución. En todos se destacan la evolución del error de la matriz de la barra completa de 81.211 elementos BARRA (curvas rojas) que presentan cierta forma funcional no lineal en la convergencia. . Evidentemente esta matriz tiene unas características distintivas con respecto a las demás que retarda la convergencia. Se observa también que el método QMR para las matrices Barra_81211 y Plenum_285265 no alcanzó la tolerancia propuesta de 10^{-14}.

Copyright © 2019 Asociación Argentina de Mecánica Computacional http://www.amcaonline.org.ar

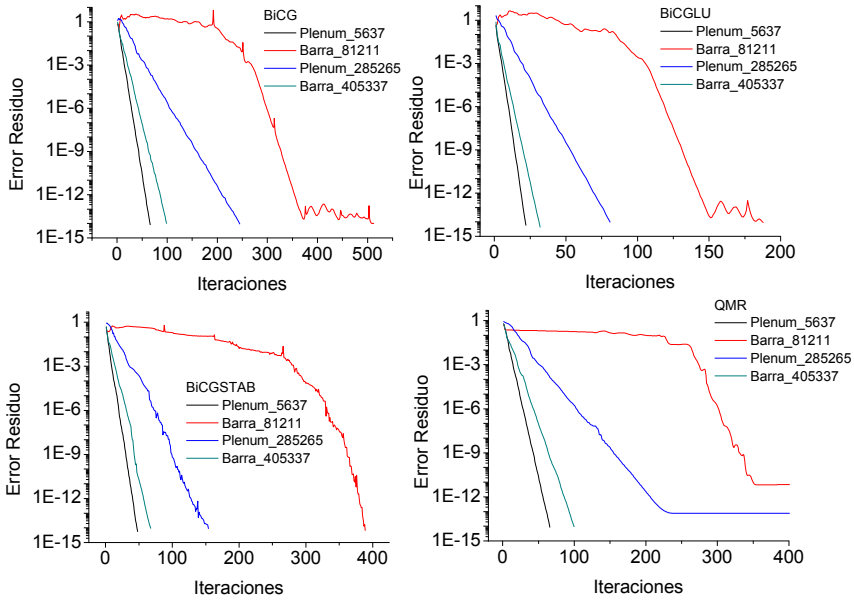

Figura 6 Convergencia de los algoritmos de resolución de las matrices dispersas agrupados por método utilizado. *Solvers*: BiCG3 (gráfico superior izquierdo), BiCGLU2 (gráfico superior derecho), BiCGSTAB (gráfico inferior izquierdo) y QMR (gráfico inferior derecho). Dominios de las matrices utilizadas: mallado con tetraedros de *plenum* de 5.637 nodos (Plenum_5637, en negro) y 285.265 nodos (Plenum_286265, en azul), y de una barra completa de 81.211 nodos (Barra_81211, en rojo) y 405.337 nodos (Barra_405337, en verde)

3.2 Problema de conducción térmica en DIONISIO

Los datos de tiempo y cantidad de iteraciones de las matrices presentadas en la Figura 2 se presentan en la Tabla 4. Se puede observar como el algoritmo de BiCGLU2 es el que presenta mejor tiempo de ejecución para la primera matriz mientras que el BiCGSTAB es el que presenta mejor tiempo de ejecución para la segunda matriz seguido por el BiCGLU2. Como el BiCGSTAB presenta el peor tiempo para la primera matriz, consideramos al BiCGLU2 como el mejor algoritmo.

	BiCG3		BiCGLU2		BiCGSTAB		QMR	
	it	tiempo	it	tiempo	it	tiempo	it	tiempo
COARSE	399	59,34	106	51,39	1180	167,74	385	85,43
TEST	124	1,41	20	1,09	81	0,90	120	1,97

Tabla 4: Tiempo de ejecución (seg) y cantidad de iteraciones de los algoritmos de resolución para las distintas matrices del problema térmico de DIONISIO

Por otro lado, la evolución de la convergencia de los *solvers* para las matrices de DIONISIO se presenta en la Figura 7. A primera vista, la matriz COARSE presenta menor estabilidad en la convergencia que la otra matriz. Además se observa para ambas matrices que el BiCGLU2 presenta menor cantidad de iteraciones para la convergencia. Para el BiCGSTAB se observa lo mismo que fue observado en la Tabla 3, que necesita más iteraciones para alcanzar la convergencia que los otros algoritmos en la primera matriz; sin embargo, para la segunda se presenta una mejora respecto a BiCG3 y QMR.

Copyright © 2019 Asociación Argentina de Mecánica Computacional http://www.amcaonline.org.ar

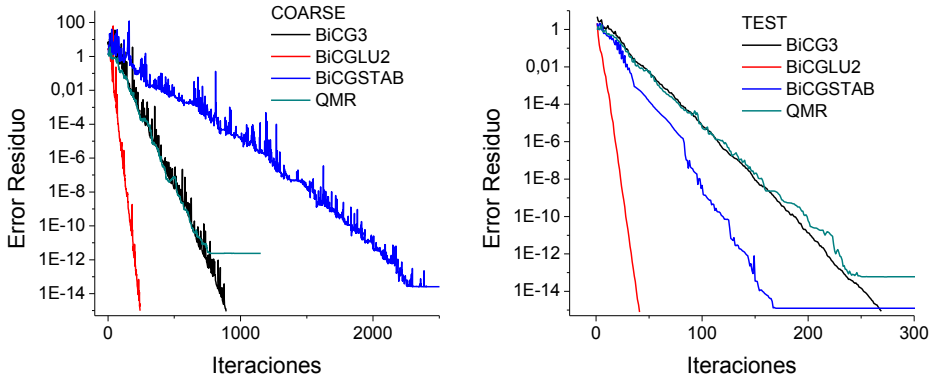

Figura 7: Convergencia de los distintos algoritmos de resolución para dos matrices extraídas de DIONISIO. *Solvers*: BiCG3 (en negro), BiCGLU2 (en rojo), BiCGSTAB (en azul) y QMR (en verde). Dominio de las matrices utilizadas: mallados con hexaedros de 311.535 nodos (COARSE) y de 23.146 nodos (TEST)

3.3 Matrices genéricas

De las ejecuciones de las matrices simétricas extraídas de MATRIX MARKET se confeccionaron gráficos de la convergencia de los distintos algoritmos. Dichos gráficos se exponen en la Figura 8. Se observa que BiCG3 converge para todas las matrices con una tolerancia de 10^{-15}. Por otro lado, el BiCGSTAB presenta una convergencia más acelerada en todos los casos aunque en ninguno de los casos alcanza la tolerancia. Por su parte, el BiCGLU2 convergió en menos de la cuarta parte de iteraciones que el resto de los algoritmos. El QMR tiene una tendencia en su convergencia similar al de BiCG3, pero en forma mucho más estable, sin oscilaciones; cabe destacar que en ciertos casos no alcanza la tolerancia impuesta (aunque siempre es menor a 10^{-8}).

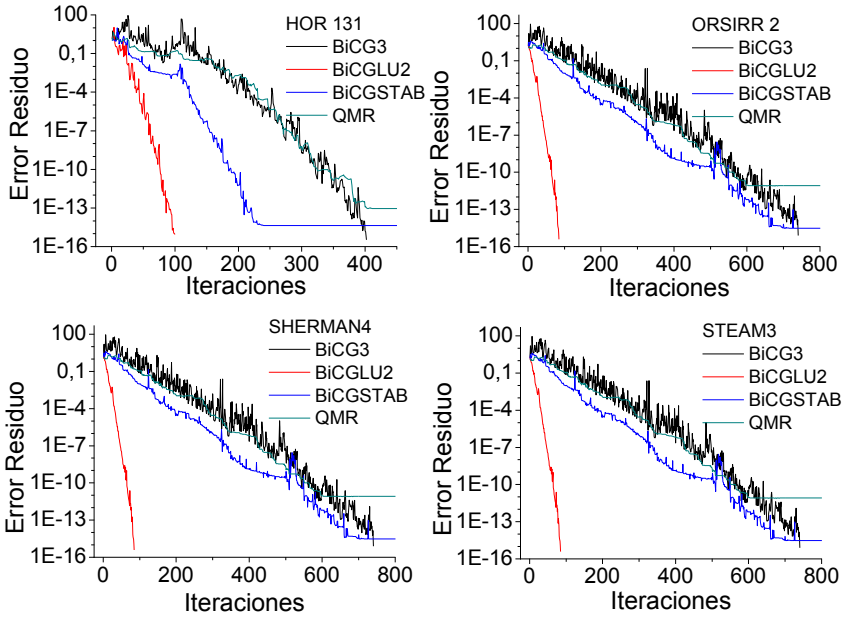

Figura 8: Convergencia de los distintos algoritmos de resolución agrupados por matriz resuelta. *Solvers*: BiCG3 (en negro), BiCGLU2 (en rojo), BiCGSTAB (en azul) y QMR (en verde). Matrices: HOR 131 (arriba izquierda), ORSIRR 2 (arriba derecha), SHERMAN4 (abajo izquierda) y STEAM3 (abajo derecha).

Copyright © 2019 Asociación Argentina de Mecánica Computacional http://www.amcaonline.org.ar

3.4 Problema de convección-difusión

A partir del análisis de las ejecuciones de los algoritmos seriales para distintas matrices se observó que el único algoritmo que convergió fue el de BiCGLU2. Esto se puede observar en el primer gráfico de la Figura 9. Por otro lado, en el segundo gráfico de la misma figura se puede observar que el error del método es menor a 10^{-60} para las tres matrices del problema. Este *solver* fue incorporado en la subrutina de resolución del problema de convección-difusión con éxito obteniendo resultados óptimos.

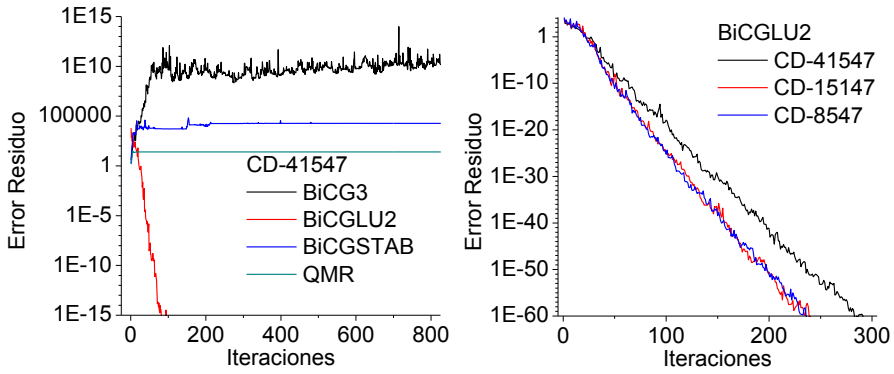

Figura 9: Gráficos de convergencia. Izquierda: Convergencia de distintos algoritmos para la matriz CD-41547. Derecha: Convergencia del algoritmo BiCGLU2 para tres matrices distintas

4 RESULTADOS DE LOS ALORITMOS PARALELIZADOS

Los tiempos obtenidos de las ejecuciones en paralelo fueron llevados a dos gráficos. Por un lado, se compararon los tiempos de ejecución de todos los algoritmos en las distintas ejecuciones y por otro se calcularon los *speedup* de las ejecuciones para analizar la eficiencia. Ambos gráficos se observan en la Figura 10. Se puede observar que el BiCG3 presenta tiempos más cortos de ejecución que el resto de los algoritmos. Por otro lado, el *speedup* muestra que dicho algoritmo no es el que mejor optimización de desempeño obtuvo pero sigue siendo aceptable para las ejecuciones.

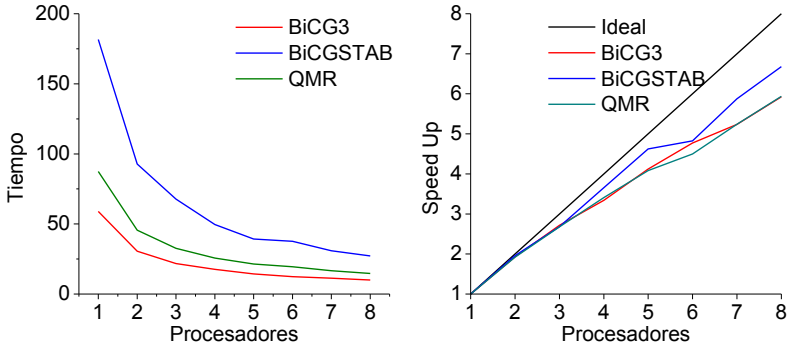

Figura 10: Tiempos de ejecución y *speedup* de los diferentes algoritmos usando diferente número de procesadores. *Solvers*: BiCG3 (en rojo), BiCGSTAB (en azul) y QMR (en verde).

Actualmente se está trabajando en paralelizar el algoritmo BiCGLU2 y analizar su desempeño en la resolución de distintos problemas para evaluar la posibilidad de incorporarlo paralelizado en el código DIONISIO.

Copyright © 2019 Asociación Argentina de Mecánica Computacional http://www.amcaonline.org.ar

5 CONCLUSIONES

Después de analizar los datos y tiempos obtenidos con los algoritmos probados, los *solvers* más eficientes corresponden al gradiente biconjugado con precondicionador ILU(0) (BiCGLU2), y el gradiente biconjugado estabilizado (BiCGSTAB). Sin embargo, el segundo mostro tiempos de ejecución elevados para casos del tipo matriz COARSE, por lo que el BiCGLU2 se convierte en la mejor opción hasta el momento. Los mismos se han añadido a DIONISIO como opción a ser elegida por los usuarios. Se han testeado con sistemas bidimensionales para comparar con los *solvers* usados tradicionalmente en el código para corroborar que su performance es adecuada en los sistemas anteriormente analizados.

Para las matrices asimétricas se observó que el BiCGLU2 presentó convergencia en menos iteraciones que el resto de los algoritmos. En el problema de convección-difusión se observó que también el *solver* BiCGLU2 es quien presenta mejor performance con respecto al resto de los algoritmos testeados.

En cuanto a los algoritmos paralelizados, se observa que el BiCG3 es el que mejor desempeño general obtuvo hasta el momento. Como trabajo a futuro se prevé paralelizar el algoritmo BiCGLU2 también de modo de optimizar su performance y obtener mejores eficiencias en multiprocesadores.

Es necesario destacar que en general, mas allá del *solver* elegido para cada caso particular, se ha incorporado al código DIONISIO una oferta extendida de *solvers* para las distintas matrices con las que debe tratar el código en alguno o varios de los problemas que resuelve.

REFERENCIAS

García León, M. D. (2003). *Estrategias para la resolución de grandes sistemas de ecuaciones lineales. Métodos de Cuasi-Mínimo Residuo Modificados.* (Tesis Doctoral, Universidad de Las Palmas de Gran Canaria). Recuperado de https://www.ana.iusiani.ulpgc.es/proyecto0507/pdf/tesislola.pdf

Goldberg, E., Loza Peralta, M. E., & Soba, A. (2019). DIONISIO 3.0: Comprehensive 3D nuclear fuel simulation through PCMI cohesive and PLENUM models. *Journal of Nuclear Materials*, *523*, 121-134. https://doi.org/10.1016/j.jnucmat.2019.06.005

Heinrich, J. C. (1980). On quadratic elements in finite element solutions of steady-state convection—diffusion equation. *International Journal for Numerical Methods in Engineering*, *15*(7), 1041-1052. https://doi.org/10.1002/nme.1620150706

Lewis, T. (2019, julio 17). Home. Recuperado 17 de julio de 2019, de OpenMP website: https://www.openmp.org/

Magoulès, F., Roux, F.-X., & Houzeaux, G. (2014). *Cálculo Científico Paralelo* (Edición: 1). Barcelona: Centro Internacional de Métodos Numéricos en Ingeniería.

Matrix Market. (2019, julio 25). Recuperado 25 de julio de 2019, de https://math.nist.gov/MatrixMarket/

Reddy, J. N. (1993). *An Introduction to the Finite Element Method* (Second Edition). Estados Unidos: McGraw-Hill.

Saad, Y. (2003). *Iterative Methods for Sparse Linear Systems: Second Edition* (Second Edition). SIAM.

Soba, A., Lemes, M., González, M. E., & Denis, A. (2016). *DIONISIO 2.0: A code to simulate the behavior of a nuclear fuel rod under irradiation in normal and accident condition.* Presentado en ENIEF 2016, Córdoba.

Copyright © 2019 Asociación Argentina de Mecánica Computacional http://www.amcaonline.org.ar