Mecánica Computacional Vol. XXIII, pp. 3211-3227
G.Buscaglia, E.Dari, O.Zamonsky (Eds.)
Bariloche, Argentina, November 2004

RESOLUCION DE GRANDES SISTEMAS DE ECUACIONES EN UN CLUSTER DE COMPUTADORAS

Victorio Sonzogni[*†], **Pablo Sanchez**[*†], **and Mario Storti**[*]

[*]Centro Internacional de Métodos Computacionales en Ingeniería (CIMEC)
INTEC, UNL - CONICET
Güemes 3450, 3000 Santa Fe, Argentina
e-mail: sonzogni@intec.unl.edu.ar, web page: http://www.cimec.org.ar

†Grupo de Investigación en Métodos Numéricos en Ingeniería (GIMNI)
Universidad Tecnológica Nacional, F. R. Santa Fe
Lavaise 610
3000 Santa Fe, Argentina

Palabras claves: computación paralela, descomposición del dominio, solución de sistemas de ecuaciones

Abstract. *En este trabajo se aborda la resolución de grandes sistemas de ecuaciones algebraicas lineales resultantes de aplicar métodos numéricos a problemas de la mecánica del continuo. La plataforma de cálculo objeto del trabajo es un cluster de microprocesadores, del tipo conocido como cluster* Beowulf. *Esta arquitectura, de memoria local, requiere normalmente una programación que contemple el intercambio de mensajes entre procesadores.*

Los métodos clásicos de resolución -directos o iterativos- poseen sus ventajas y desventajas. Las desventajas de los métodos directos cuando el problema es muy grande lo limitan para ser aplicados como único resolvedor para el tipo de problemas en estudio, dejando camino para los procedimientos iterativos como alternativa práctica. Sin embargo éstos últimos tampoco se desempeñan satisfactoriamente para estos problemas grandes. Hay técnicas basadas en particionar el dominio y efectuar resoluciones a niveles de las incógnitas de cada subdominio y a nivel de aquellas en las interfases entre subdominios. Son las técnicas de descomposición de dominio que combinan resoluciones directas e iterativas. Se requiere adecuados precondicionadores a fin de obtener resultados satisfactorios.

En este trabajo se muestran resultados obtenidos en aplicaciones y se discute el comportamiento de diversos algoritmos para resolución, así como de diversos precondiconadores.

1. INTRODUCCION

La resolución de grandes sistemas de ecuaciones algebraicas lineales subyace en la solución numérica de problemas de la mecánica del continuo y muchos otros problemas ingenieriles, llegando a constituir en muchos casos el principal factor de costo computacional.

Entre las plataformas de cálculo, los *clusters* de microprocesadores han resultado ser una alternativa muy eficiente y abordable para resolver grandes problemas numéricos. En el CIMEC, existe un cluster de tipo Beowulf[1] con 20 nodos Pentium IV, los nodos más recientes de 2.8GHz y 2GB de memoria RAM.

Los métodos clásicos de resolución se suelen clasificar en *directos* o *iterativos*. Los primeros proporcionan una solución cerrada pero el principal inconveniente es el costo tanto en tiempo de procesamiento como en almacenamiento en memoria. La cantidad de operaciones requeridas para una matriz llena es del orden de n^3, siendo n la cantidad de incógnitas. Los procedimientos iterativos son entonces preferidos para resolver sistemas grandes. Sin embargo en grandes sistemas de ecuaciones el condicionamiento de la matriz empeora y la solución se dificulta. En la solución iterativa se hace necesario precondicionar satisfactoriamente la matriz del sistema de modo de poder obtener la solución en un número razonable de iteraciones. Aún así se han encontrado casos en que la solución directamente no es alcanzable.

Hay técnicas basadas en particionar el dominio y efectuar resoluciones a niveles de las incógnitas de cada subdominio y a nivel de aquellas en la interfaz entre subdominios. Son las técnicas de descomposición de dominio que combinan resoluciones directas e iterativas. Se requiere adecuados precondicionadores a fin de obtener resultados satisfactorios.

En este trabajo se presentan los métodos clásicos directos e iterativos, y los métodos basados en descomposición del dominio. Se indican también procedimientos para precondicionamiento. Finalmente se muestran algunos resultados obtenidos con distintas configuraciones de cluster a lo largo del tiempo, pero que ilustran el desempeño de las distintas alternativas para resolución.

2. METODOS DE SOLUCION DE SISTEMAS DE ECUACIONES ALGEBRAICAS LINEALES

Los métodos de resolución de sistemas de ecuaciones lineales se clasifican en *directos* e *iterativos*.

2.1. Métodos directos

Los métodos directos se basan en efectuar una transformación del sistema en otro equivalente, esto es que posee el mismo vector solución, mediante transformaciones elementales tales como intercambiar ecuaciones o realizar una combinación lineal de dos de ellas. Se busca de esta forma obtener sistemas equivalentes cuya solución sea más fácil. Así la resolución del sistema

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{1}$$

Se realiza en los siguientes pasos:

1. Factorización de la matriz $\mathbf{A}$

   esto es, encontrar la matrices $\mathbf{L}$ y $\mathbf{U}$ tales que:

$$\mathbf{A} = \mathbf{LU} \tag{2}$$

   donde $\mathbf{L}$ es una matriz triangular inferior y $\mathbf{U}$ una triangular superior.

2. Hallada la desomposición precedente, resolver

$$\mathbf{Ly} = \mathbf{b} \tag{3}$$

3. Conocido el vector $\mathbf{y}$ resolver

$$\mathbf{Ux} = \mathbf{y} \tag{4}$$

La decomposición LU de una matriz no es única. Una descomposición en la que los términos de la diagonal de $\mathbf{L}$ son unitarios ($l_{ii} = 1$) se conoce con el nombre de *Doolittle*. Por otra parte, si son unitarios los coeficientes de la diagonal de $\mathbf{U}$ ($u_{ii} = 1$) se habla de factorización de *Crout*. En el caso de matrices simétricas, definidas positivas, una descomposición popular es la de *Cholesky* donde $\mathbf{U} = \mathbf{L^T}$.

Las operaciones básicas a realizar con métodos directos de resolución pueden ser de tipo SAXPY (Sum of Alpha X Plus Y) o producto internos de vectores. La ejecución en paralelo de estos algoritmos se puede efectuar por el denominado *paralelismo algebraico*. Aquí el mismo algoritmo que se plantea secuencialmente, cuando se hacen los cálculos se descompone entre los procesadores. Esto exige la distribución acorde de los datos entre los distintos nodos y plantea diversas posibilidades de distribución.

Los métodos directos son caros en términos de operaciones. La factorization requiere $O(n^3)$ operaciones, mientras que las sustituciones hacia adelante y hacia atrás, requeridas para obtener la solución, se efectúan en $O(n^2)$ operaciones.

También son exigentes en lo que respecta a almacenamiento de variables ya que se precisa disponer de la matriz, pudiendo en ésta sobre-escribirse los factores L y U. Este requerimiento de almacenamiento, al igual que la cantidad de operaciones a realizar, pueden verse favorablemente reducidos si la matriz tiene estructura rala, banda o si ella es simétrica.

2.2. Métodos iterativos

Los métodos iterativos se basan en construir una secuencia de soluciones aproximadas $\{\mathbf{x}^k\}$ $(k = 1, 2, \ldots)$ tal que cuando el índice de la iteración $k \to \infty$ converja a la solución del sistema de ecuaciones.

Las fórmulas de recurrencia pueden presentarse de diferentes maneras. Por ejemplo se puede plantear

$$\mathbf{x}^k = \mathbf{G}\mathbf{x}^{k-1} + \mathbf{c} \tag{5}$$

Cuadro 1: Algoritmo del método del gradiente conjugado

A.	Initialización	
A.1	$\mathbf{x}$	`estimación inicial`
A.2	$\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$	`matriz` $\times$ `vector + suma vect.`
A.3	$\mathbf{p} = \mathbf{r}$	
A.4	$\rho = (\mathbf{r}, \mathbf{p})$	`producto interno`
A.5	$\rho_0 = \rho$	
A.6	$k = 1$	
B.	Iteraciones. Mientras $k < Kmax$ realiza:	
B.1	Test de convergencia: si $\rho < Tol\ \rho_0$ termina	
B.2	$\mathbf{a} = \mathbf{A}\mathbf{p}$	`matriz` $\times$ `vector`
B.3	$m = (\mathbf{p}, \mathbf{a})$	`producto interno`
B.4	$\alpha = \frac{\rho}{m}$	
B.5	$\mathbf{x} = \mathbf{x} + \alpha\mathbf{p}$	SAXPY
B.6	$\mathbf{r} = \mathbf{r} - \alpha\mathbf{a}$	SAXPY
B.7	$\rho_{old} = \rho$	
B.8	$\rho = (\mathbf{r}, \mathbf{r})$	`producto interno`
B.9	$\gamma = \frac{\rho}{\rho_{old}}$	
B.10	$\mathbf{p} = \mathbf{r} + \gamma\mathbf{p}$	SAXPY
B.11	$k = k + 1$, go to B.1	

que a partir de una estimación inicial $\mathbf{x}^0$ permita obtener la solución aproximada en cada iteración. Para que el método conveja se requiere que $||\mathbf{G}|| < 1$. Clásicos métodos iterativos como Jacobi, Gauss-Seidel o SOR pueden encuadrarse en esta tipología.

Procedimientos basados en optimización resultan eficientes para la resolución iterativa de sistemas de ecuaciones. Entre ellos, el método de gradientes conjugados (para matrices simétricas) o GMRES (para matrices no simétricas) gozan de extendida popularidad. En ellos las iteraciones se realizan en la forma

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k\ \mathbf{p}^k \tag{6}$$

a partir de una estimación inicial $\mathbf{x}^0$.

En cada paso es preciso calcular una dirección de búsqueda $\mathbf{p}^k$, con fórmulas propias de cada método, y un paso de avance α^k en esa dirección.

El algoritmo del método de gradiente conjugado se indica en el Cuadro I.

Estos métodos, basados en iteración por subespacios, poseen buenas propiedades de convergencia. No obstante, si el número de condición es alto (lo que típicamente sucede en grandes

problemas) es necesario introducir un precondicionamiento. Esto se discutirá en secciones siguientes.

En el algoritmo de Cuadro I se indica el tipo de operación a realizar. Los cálculos más pesados son los relativos a producto de matriz por vector. También es necesario realizar producto interno de vectores y actualización de vectores (SAXPY).

Si las matrices se distribuyen adecuadamente entre los procesadores, por bloques de filas, el producto matriz por vector puede realizarse con independencia en cada procesador, lo mismo que la actualizacón de vectores. El producto interno, sin embargo, requiere comunicación para realizar la operación de reducción. Y el posterior envío del escalar resultante a todos los procesadores.

La matriz del sistema de ecuaciones interviene en el proceso solamente al realizar el producto matriz por vector. Es importante resaltar que no se precisa disponer de la matriz global en ningún momento. Para efectuar el producto matriz por vector podrían utilizarse las matrices elementales del método de elementos finitos y ensamblarse el vector resultante.

Para grandes sistemas de ecuaciones, en que los métodos directos resultan prohibitivos, los métodos iterativos son a menudo utilizados. Si las matrices poseen número de condición no muy alto, estos últimos procedimientos pueden resultar satisfactorios. Por otra parte la solución iterativa contiene un error algorítmico. El costo de la solución puede ajustarse según la tolerancia especificada para el problema.

3. METODOS DE DESCOMPOSICION DEL DOMINIO

Estos métodos se basan en descomponer el dominio de definición del problema en subdominios de modo que en cada uno de estos el problema sea más fácil de resolver (por ejemplo si se lo resuelve analíticamente), o bien sea de un tamaño adecuado para ser alojado en un procesador. Estos subdominios pueden solaparse (método de Schwarz) o no. Entre los que no se solapan, es decir el contacto se produce únicamente en las fronteras inter-subdominio, se incluyen los procedimientos que utilizan el complemento de Schur.

3.1. Complemento de Schur

Considérese una descomposición como en la figura 1. Con Ω^s se designa el subdominio s ($s = 1, NSD$) y con Γ_i^s $(i = 1, 3)$ las fronteras del mismo. Γ_1 se utilirará para indicar el contorno con condiciones de tipo Dirichlet, Γ_2 para aquel con condiciones de tipo Neumann y Γ_3 para las fronteras con otros subdominios.

Si se utiliza el método de los elementos finitos, para resolver numéricamente el problema mecánico se llega a un sistema de ecuacions

$$\mathbf{K}\mathbf{u} = \mathbf{f} \tag{7}$$

siendo $\mathbf{u}$ el vector de las variables primales nodales (desplazamientos, en un problema de sólido deformable elástico) ; $\mathbf{f}$ es el vector de variables discretas duales (fuerzas nodales); y $\mathbf{K}$ la matriz del sistema (rigidez).

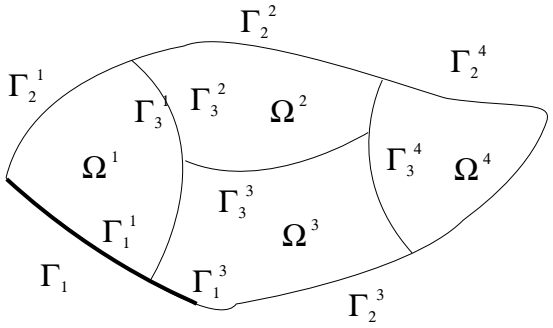

Figura 1: Descomposición del dominio

Las matrices de la ecuación (7) se construyen por subdominios y aquellas restringidas al subdomino s se designan por: $\mathbf{K^s}$, $\mathbf{u^s}$ y $\mathbf{f^s}$. Se pueden particionar en grupos de incógnitas (grados de libertad) internos al subdominio $\mathring{\mathbf{u}}^s$ y aquellos en la interfaz $\bar{\mathbf{u}}^\mathbf{s}$. La matriz de rigidez se puede así escribir:

$$\mathbf{K}^s = \begin{bmatrix} \mathring{\mathbf{K}}^s & \tilde{\mathbf{K}}^\mathbf{s} \\ \tilde{\mathbf{K}}^{\mathbf{s},\mathbf{T}} & \bar{\mathbf{K}}^\mathbf{s} \end{bmatrix} \tag{8}$$

y los vectores de desplazamientos y fuerzas nodales:

$$\mathbf{u}^s = \begin{bmatrix} \mathring{\mathbf{u}}^s \\ \bar{\mathbf{u}}^s \end{bmatrix} \quad \text{y} \quad \mathbf{f}^s = \begin{bmatrix} \mathring{\mathbf{f}}^s \\ \bar{\mathbf{f}}^s \end{bmatrix} \tag{9}$$

El símbolo $\mathring{\square}$ se usa aquí para grados de libertad *internos*, $\bar{\square}$ para aquellos en la *interfaz* y $\vec{\square}$ para la interacción entre ambos.

Si, por otra parte, se ensambla la contribución de todos los subdominios a los grados de interfaz globales, se puede escribir

$$\mathbf{K}_I = \mathop{\mathbf{A}}_{s=1}^{NSD} \bar{\mathbf{K}}^s \tag{10}$$

siendo $\mathbf{u}_I$ el vector de desplazamientos en grados de libertad de interfaz de todo el dominio. Las matrices con subíndice I tienen el tamaño del problema de interfaz global.

Volviendo ahora sobre el problema completo, reordenando sus variables, la matriz de rigidez se puede escribir:

$$\mathbf{K} = \begin{bmatrix} \mathring{\mathbf{K}}^1 & 0 & \dots & 0 & \dots & \vec{\mathbf{K}}_I^1 \\ 0 & \mathring{\mathbf{K}}^2 & \dots & 0 & \dots & \vec{\mathbf{K}}_I^2 \\ \vdots & \vdots & \ddots & \vdots & & \vdots \\ 0 & 0 & \dots & \mathring{\mathbf{K}}^s & \dots & \vec{\mathbf{K}}_I^s \\ \vdots & \vdots & & \vdots & \ddots & \vdots \\ \vec{\mathbf{K}}_I^{1,T} & \vec{\mathbf{K}}_I^{2,T} & \dots & \vec{\mathbf{K}}_I^{s,T} & \dots & \mathbf{K}_I \end{bmatrix} \tag{11}$$

y los vectores

$$\mathbf{u} = \begin{bmatrix} \mathring{\mathbf{u}}^1 \\ \mathring{\mathbf{u}}^2 \\ \cdots \\ \mathring{\mathbf{u}}^s \\ \cdots \\ \mathbf{u}_I \end{bmatrix} \quad \text{and} \quad \mathbf{f} = \begin{bmatrix} \mathring{\mathbf{f}}^1 \\ \mathring{\mathbf{f}}^2 \\ \cdots \\ \mathring{\mathbf{f}}^s \\ \cdots \\ \mathbf{f}_I \end{bmatrix} \tag{12}$$

La ecuación de equilibrio (7) se puede particionar en los siguientes sistemas:

$$\begin{cases} \mathring{\mathbf{K}}^s \mathring{\mathbf{u}}^s + \vec{\mathbf{K}}_I^s \mathbf{u}_I \;=\; \mathring{\mathbf{f}}^s & , s = 1, NSD \\ \displaystyle\sum_{s=1}^{NSD} \vec{\mathbf{K}}_I^{s,T} \mathring{\mathbf{u}}^s \;+\; \mathbf{K}_I \mathbf{u}_I \;=\; \mathbf{f}_I \end{cases} \tag{13}$$

Efectuando eliminación gaussiana por bloques:

$$\begin{cases} \mathring{\mathbf{K}}^s \mathring{\mathbf{u}}^s \;=\; \mathring{\mathbf{f}}^s - \vec{\mathbf{K}}_I^s \mathbf{u}_I & , s = 1, NSD \\ [\mathbf{K}_I - \displaystyle\sum_{s=1}^{NSD} \vec{\mathbf{K}}_I^{s,T} (\mathring{\mathbf{K}}^s)^{-1} \vec{\mathbf{K}}_I^s] \mathbf{u}_I \;=\; \mathbf{f}_I - \displaystyle\sum_{s=1}^{NSD} \vec{\mathbf{K}}_I^{s,T} (\mathring{\mathbf{K}}^s)^{-1} \mathring{\mathbf{f}}^s \end{cases} \tag{14}$$

La matriz de la segunda ecuación en (14):

$$\mathbf{S} \;=\; \mathbf{K}_I - \sum_{s=1}^{NSD} \vec{\mathbf{K}}_I^{s,T} (\mathring{\mathbf{K}}^s)^{-1} \vec{\mathbf{K}}_I^s \tag{15}$$

se conoce como *complemento de Schur* o matriz de *capacitancia*.

El primer grupo de ecuaciones en (14) representa el sistema asociado a los grados de libertad *internos* $\mathring{\mathbf{u}}^s$ a cada subdominio, resultantes de la característica de no penetración de la descomposición. Esta parte de la solución es perfectamente paralelizable. El tamaño de estos problemas está dado por la granularidad de la descomposición en subdominios.

La segunda parte de (14) representa el problema de *interfaz*. El tamaño de la matriz del complemento de Schur $\mathbf{S}$ es bastante menor que el de la matriz global $\mathbf{K}$, pero $\mathbf{S}$ es densa. El número de condición de $\mathbf{S}$ es también bastante menor que el de $\mathbf{K}$. Por otra parte no es necesario el ensamble explícito de la matriz $\mathbf{S}$, pudiendo efectuarse las operaciones para la resolución, por subdominios. Esta parte de la solución es acoplada para todo el problema requiriendo comunicación entre los distintos procesadores para su resolución en paralelo.

La solución del problema (14) puede verse como efectuada en dos partes: un problema de grados de libertad de interfaz y otro de grados de libertad internos a los subdominios. Se suele resolver el problema interno a través de métodos directos y el problema en la interfaz mediante técnicas iterativas. El uso de métodos directos para los problemas internos evita errores algorítmicos que se propaguen al problema de interfaz. Como el tamaño de los problemas internos es acotado al subdominio los métodos directos son aplicables.

Para el problema de interfaz, sin embargo, un método directo no es atractivo, por los requerimientos de almacenamiento. La matriz $\mathbf{S}$ es una matriz completa y cara para construir. Por este motivo se suele recurrir a métodos iterativos (Gradiente conjugado, GMRES) con un adecuado precondicionamiento. La descomposición de dominio brinda un adecuado precondicionamiento para el problema global.

3.2. Solución iterativa del problema de interfaz

La sub-matriz de rigidez asociada a los grados de libertad de interfaz $\mathbf{K}_I$ puede escribirse

$$\mathbf{K}_I \;=\; \sum_{s=1}^{NSD} \mathbf{K}_I^s \tag{16}$$

y el complemento de Schur

$$\mathbf{S} \;=\; \sum_{s=1}^{NSD} \mathbf{S}^s \tag{17}$$

donde

$$\mathbf{S}^s \;=\; \mathbf{K}_I^s - \vec{\mathbf{K}}_I^{s,T} \left(\mathring{\mathbf{K}}^s\right)^{-1} \vec{\mathbf{K}}_I^s \tag{18}$$

La ecuación (17) muestra que la contribución de cada subdominio a la matriz $\mathbf{S}$ puede calcularse independientemente. La ecuación (14-b) puede ser re-escrita:

$$\mathbf{S}\,\mathbf{u}_I \;=\; \mathbf{g}_I \tag{19}$$

La solución de la ecuación (19) por un método iterativo (gradiente conjugado para este caso de elasticidad lineal) se puede realizar con un algoritmo como el del Cuadro II. Allí se considera un sistema genérico

$$\mathbf{A}\mathbf{x} \;=\; \mathbf{b} \tag{20}$$

y se realiza además un *precondicionamiento* con la intención de bajar el número de condición de la matriz.

Puede observarse que las fases del proceso que resultan más demandantes en tiempo de procesamiento son el producto matriz-vector en los pasos A.2 y B.1, y la solución del sistema de ecuaciones implícita en el precondicionamiento, en los pasos A.3 y B.7. Las restantes operaciones sobre vectores, del tamaño del problema de interfaz global, son productos internos y actualización de vectores (SAXPY).

3.3. Precondicionamiento

El precondicionamiento resulta indispensable para la resolución de grandes sistemas de ecuaciones por técnicas iterativas. Existen varios procedimientos para precondicionamiento. Entre ellos puede mencionarse:

Cuadro 2: Algoritmo del gradiente conjugado preconditionado

A.	Initialization	
A.1	$\mathbf{x}$	`estimac. inicial`
A.2	$\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$	`matriz` $\times$ `vector + suma vect.`
A.3	solve $\mathbf{P}\mathbf{z} = \mathbf{r}$	`solución sistema`
A.4	$\rho = (\mathbf{r}, \mathbf{z})$	`producto interno`
A.5	$\rho_0 = \rho$	
A.6	$\mathbf{p} = \mathbf{z}$	
A.7	$k = 1$	
B.	Iterations: while $k < Kmax$ do	
B.1	Convergence test: if $\rho < Tol\ \rho_0$ end iterations	
B.2	$\mathbf{a} = \mathbf{A}\mathbf{p}$	`matriz` $\times$ `vector`
B.3	$m = (\mathbf{p}, \mathbf{a})$	`producto interno`
B.4	$\alpha = \frac{\rho}{m}$	
B.5	$\mathbf{x} = \mathbf{x} + \alpha\mathbf{p}$	`SAXPY`
B.6	$\mathbf{r} = \mathbf{r} - \alpha\mathbf{a}$	`SAXPY`
B.7	solve $\mathbf{P}\mathbf{z} = \mathbf{r}$	`solución sistema`
B.8	$\rho_{old} = \rho$	
B.9	$\rho = (\mathbf{r}, \mathbf{z})$	`producto interno`
B.10	$\gamma = \frac{\rho}{\rho_{old}}$	
B.11	$\mathbf{p} = \mathbf{z} + \gamma\mathbf{p}$	`SAXPY`
B.12	$k = k + 1$, go to B.1	

1. Jacobi o escalado diagonal

   Este es el más simple y se basa en tomar como precondicionador la matriz

   $$\mathbf{P} = \mathrm{diag}(\mathbf{A}) \tag{21}$$

   No requiere resolver sistema alguno, pero solamente es eficiente si la matriz del sistema es diagonal dominante.

2. Factorización incompleta

   En este caso la factorización $\mathbf{LU}$, o la factorización de Cholesky $\mathbf{C^T C}$ para el caso de matrices simétricas positiva definidas, se efectúa pero deteniendo el proceso al alcanzar la estructura rala de la matriz A. Se propone entonces como precondicionador:

$$\mathbf{P} = \tilde{\mathbf{L}}^{\mathbf{T}}\tilde{\mathbf{U}} \tag{22}$$

siendo $\tilde{\mathbf{L}}$ y $\tilde{\mathbf{U}}$ los factores incompletos. Este precondicionador podría llevar a tener pivotes nulos. Se han introducido modificaciones tales como *Shifted Incomplete Cholesky Factorization* (Manteuffel, 1979)[9] para eliminar este problema.

3. Element by Element (EBE) (Winget and Hughes, 1985)[18]

   Utilizando dos descomposiciones, aditiva y multiplicativa, de la matriz este método construye un precondicionador a partir de las matrices elementales.

El método de descomposición de dominio puede mirarse como un buen procedimiento para precondicionar el problema *global*. Como casos límites se comporta como un procedimiento directo cuando el tamaño de los subdominios tiende a uno, o como un método iterativo global con un precondicionador tipo EBE cuando la cantidad de subdominios tiende a la cantidad de elementos. Se han reportado resultados que muestran al método de descomposición de dominio, o un precondicionador *subdominio-por-subdominio*, como más eficiente que uno elemento-por-elemento.[10]

3.4. **Método Neuman-Neuman** (Bjørstad and Widlund, 1986)[2]

Si se observa el algoritmo del Cuadro II se puede ver que las partes que demandan mayor tiempo de procesamiento son, como se ha indicado, el producto matriz-vector en los pasos A.2 y B.2, y la solución del sistema de ecuaciones implícita en el precondicionamiento, en los pasos A.3 y B.7.

El producto matriz por vector (A.2 y B.2) puede escribirse:

$$\mathbf{a} = \mathbf{S}\mathbf{p} \tag{23}$$

siendo $\mathbf{S}$ el complemento de Schur y, debido a (17),

$$\mathbf{a} = \sum_{s=1}^{NSD} \mathbf{a}^s = \sum_{s=1}^{NSD} \mathbf{S}^s\mathbf{p} \tag{24}$$

esto es, las contribuciones a $\mathbf{a}$ se calculan separadamente en cada subdominio. En el subdominio s se tiene: (18)

$$\mathbf{S}^s\mathbf{p} = [\mathbf{K}_I^s - \vec{\mathbf{K}}_I^{s,T}(\mathring{\mathbf{K}}^s)^{-1}\vec{\mathbf{K}}_I^s]\mathbf{p} \tag{25}$$

y considerando el problema restringido al subdominio:

$$\begin{bmatrix} \mathring{\mathbf{K}}^s & \vec{\mathbf{K}}^s \\ \vec{\mathbf{K}}^{s,T} & \bar{\mathbf{K}}_I^s \end{bmatrix} \begin{bmatrix} \mathbf{v}^s \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{a}^s \end{bmatrix} \tag{26}$$

La contribución del subdominio s al vector (24) es

$$\mathbf{a}^s \;=\; \vec{\mathbf{K}}^{s,T}\mathbf{v}^s + \mathbf{K}_i^s\mathbf{p} \tag{27}$$

siendo $\mathbf{v}^s$ solución de

$$\mathring{\mathbf{K}}^s\mathbf{v}^s = -\vec{\mathbf{K}}^s\mathbf{p} \tag{28}$$

Las ecuaciones (26) a (28) muestran que para evaluar el producto matriz-vector (24) basta con resolver, en cada subdominio, un problema de *Dirichlet* donde valores prescriptos de $\mathbf{p}$ se impone en la interfaz Γ_3^s, y se obtiene el vector asociado $\mathbf{a}^s$. Finalmente se suman las contribuciones $\mathbf{a}^s$ de cada subdominio.

Para realizar el precondicionamiento se efectúa el siguiente procedimiento (De Roeck and Le Tallec, 1991).[5] En cada subdominio se define una matriz $\mathbf{D}_I^s$ de modo que

$$\sum_{s=1}^{NSD} \mathbf{D}_I^s \;=\; \mathbf{I}_I \tag{29}$$

Esto significa que ensamblando las matrices $\mathbf{D}_I^s$ de todos los subdominios se obtiene la matriz identidad en el espacio de interfaz global. La manera más simple de construir $\mathbf{D}_I^s$ es con una matriz diagonal cuyos términos son la inversa de la cantidad de subdominios que comparten el grado de libertad en cuestión.

En cada iteración del gradiente conjugado, el residuo se proyecta sobre cada subdominio:

$$\mathbf{r}^s \;=\; \mathbf{D}^{s,T}\mathbf{r} \tag{30}$$

En cada subdominio se resuelve el sistema:

$$\mathbf{S}^s\mathbf{z}^s \;=\; \mathbf{r}^s \tag{31}$$

Finalmente las contribuciones de cada subdominio al vector z se promedian sobre la interfaz:

$$\mathbf{z} \;=\; \sum_{s=1}^{NSD} \mathbf{D}^s\mathbf{z}^s \tag{32}$$

Esto es equivalente a usar un precondicionamiento de la forma:

$$\mathbf{P}^{-1} \;=\; \sum_{s=1}^{NSD} \mathbf{D}^s(\mathbf{S}^s)^{-1}\mathbf{D}^{sT} \tag{33}$$

La solución de (31), a su vez, es equivalente a resolver el problema de *Neumann* en el subdominio s, donde el vector solución $\mathbf{z}^s$ contiene, en un problema elástico, desplazamientos de los grados de libertad de interfaz. Puede ser realizado sin formar explícitamente la matriz del complememto de Schur $\mathbf{S}$, escribiendo para cada subdominio:

$$\begin{bmatrix} \mathring{\mathbf{K}}^s & \tilde{\mathbf{K}}^{\mathbf{s}} \\ \tilde{\mathbf{K}}^{\mathbf{s},\mathbf{T}} & \bar{\mathbf{K}}_I^s \end{bmatrix} \begin{bmatrix} \mathbf{v}^s \\ \mathbf{z}^s \end{bmatrix} \;=\; \begin{bmatrix} 0 \\ \mathbf{r}^s \end{bmatrix} \tag{34}$$

La solución del problema de Neumann 34 en el sundominio s es

$$\mathbf{v}^s \;=\; -(\overset{\circ}{\mathbf{K}}{}^s)^{-1}\vec{\mathbf{K}}^s\mathbf{z}^s \tag{35}$$

$$\mathbf{z}^s \;=\; \left(\mathbf{K}_I^s - \vec{\mathbf{K}}^{s,T}(\overset{\circ}{\mathbf{K}}{}^s)^{-1}\vec{\mathbf{K}}^s\right)^{-1}\mathbf{r}^s \;=\; \left(\mathbf{S}^s\right)^{-1}\mathbf{r}^s \tag{36}$$

Cuando se resuelve iterativamente el problema de interfaz el producto matriz-vector (A.2 y B.2) y la solución del sistema de ecuaciones (A.3 y B.7) se reemplazan por resoluciones del problema restricto a cada subdominio, alternativamente con condiciones de Dirichlet o de Neumann, respectivamente. Estas soluciones por subdominio son perfectamente paralelizables.

Para un problema de Laplace mientras el número de condición de la matriz global is $O(\frac{1}{h^2})$ (siendo h el tamaño de los elementos) aquel para el complemento de Schur es $O(\frac{1}{h})$, y utilizando el precondicionador (33) se reduce a $O(1)$.

En general, en un problema de elasticidad, la ecuación (31), sobre un subdominio, posee a una matriz $\mathbf{S}$ singular a menos que se restrinjan suficientes grados de libertad para impedir movimientos de cuerpo rígido. Diversas técnicas han sido desarrolladas para eliminar este inconveniente en los subdominios "flotantes".[3,5–8]

3.5. Precondicionador ISP (*Interface Strip*)(Storti et al., 2002)

El precondicionador *Interface Strip* propuesto por Storti et al[15] está basado en resolver, alrededor de la interfaz un problema sobre una franja de elementos. En este sentido mejora el desempeño de procedimiento Neumann-Neumann, que *distribuye* las fuerzas de desequilibrio (flujo, en un problema térmico) entre los subdominios, resuelve el problema en cada uno y finalmente *promedia* los desplazamientos (temperaturas) resultantes de interfaz. Los resultados obtenidos confirman la mayor eficiencia de este precondicionador frente al Neumann-Neumann o a la resolución iterativa global.[15]

4. ALGUNOS RESULTADOS

En esta sección se muestran algunos resultados de problemas resueltos en el cluster Gerónimo , de tipo Beowulf, del CIMEC.[4] Se ha utilizado el software PetscFem,[16] que utiliza la biblioteca Petsc.[11]

En general el desempeño cualitativo de estos métodos es como se indica en la figura 2. Los recursos necesarios aumentan al achicar la tolerancia para finalizar el proceso iterativo del resolvedor. Pero este aumento se hace más crítico para resolvedores globales, a medida que se toman tolerancias muy pequeñas. Para los métodos directos, claro está, no se modifican los requerimientos con la tolerancia.

La figura 3 muestra resultados de convergencia obtenidos en un problema de respuesta mecánica elástica incompresible con 56000 elementos tetraédros mixtos[12] . Las curvas allí se refieren a una resolución iterativa del problema global (GMRES) y a la solución con descomposición del dominio, con precondicionador para el problema de interfaz de tipo Jacobi o IS. El mismo tipo de resultados se observa en la figura 4 pero esta vez para una malla de 90000 elementos.

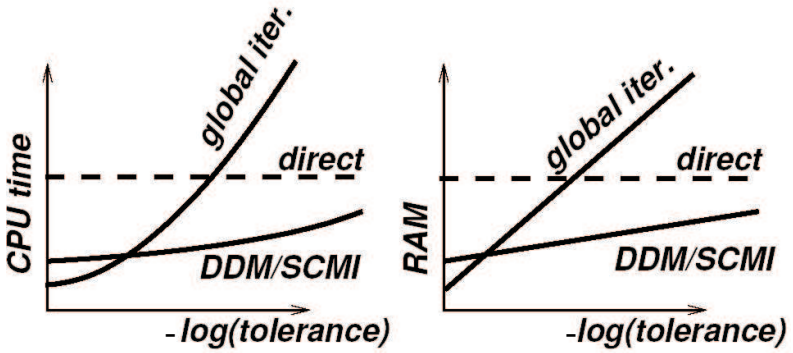

Figura 2: Desempeño de métodos iterativos y de descomposición de dominio

En los cuadros 3 y 4 figura el tiempo de resolución de estos mismos problemas.

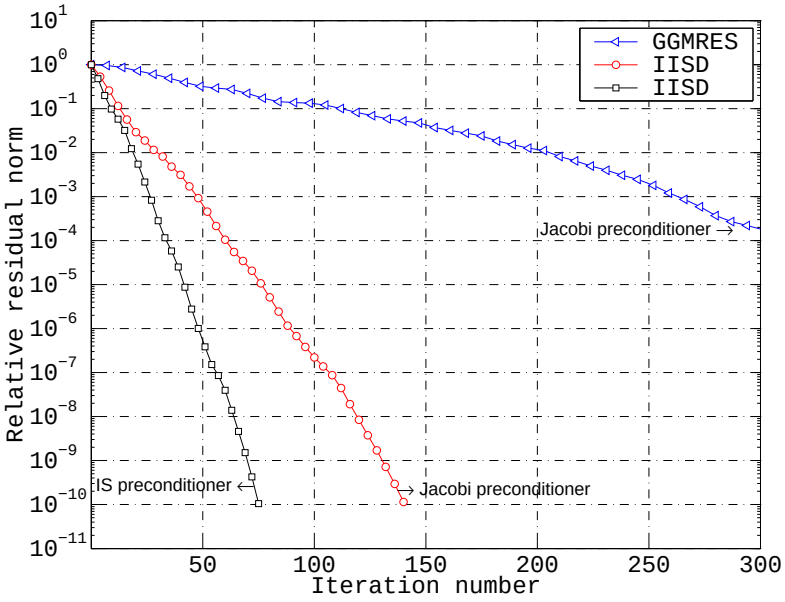

Figura 3: Linear solver convergence, test with 56000 elements.

La convergencia en un problema de flujo de Navier Stokes, en una cavidad cúbica, con 625000 elementos se muestra en la figura 5, para un método iterativo global y uno de descomposición de dominio. Cabe mencionar que este caso fue calculado en 8 procesadores P4 1.7 Ghz con 512MB RIMM (Rambus). Para un tamaño mayor del problema (2.2x10^6 elementos) a las 122 iteraciones la iteración global produjo un agotamiento de memoria. La solución por descomposición de dominio pudo en este caso obtenerse.

En la figura 6 se muestran resultados de un problema de Poisson con malla de 120×120 elementos.[15] Allí se pueden ver las características de convergencia de un método iterativo global (GMRES) y de descomposición del dominio con precondicionamiento Neumann-Neumann (N-N)y Interface Strip (IS), para diferentes anchos de la banda. El precondicionador N-N es el

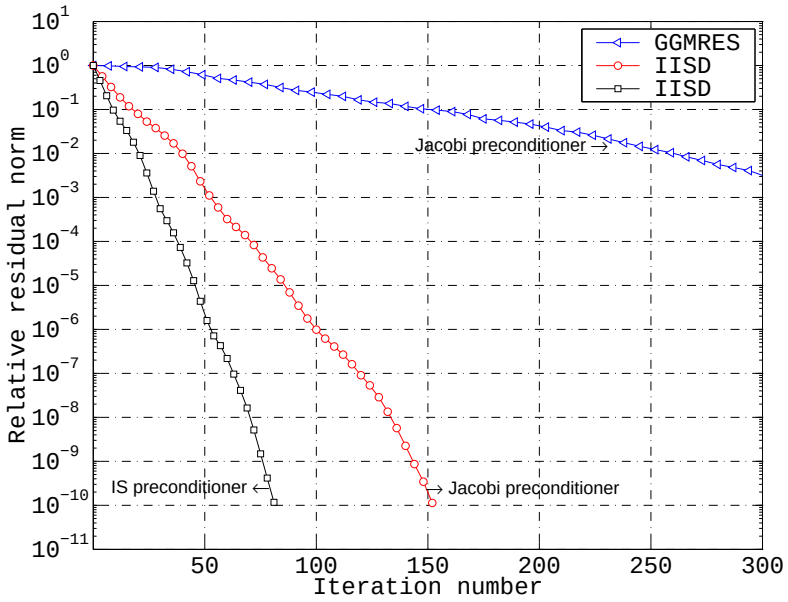

Figura 4: Linear solver convergence, test with 90000 elements.

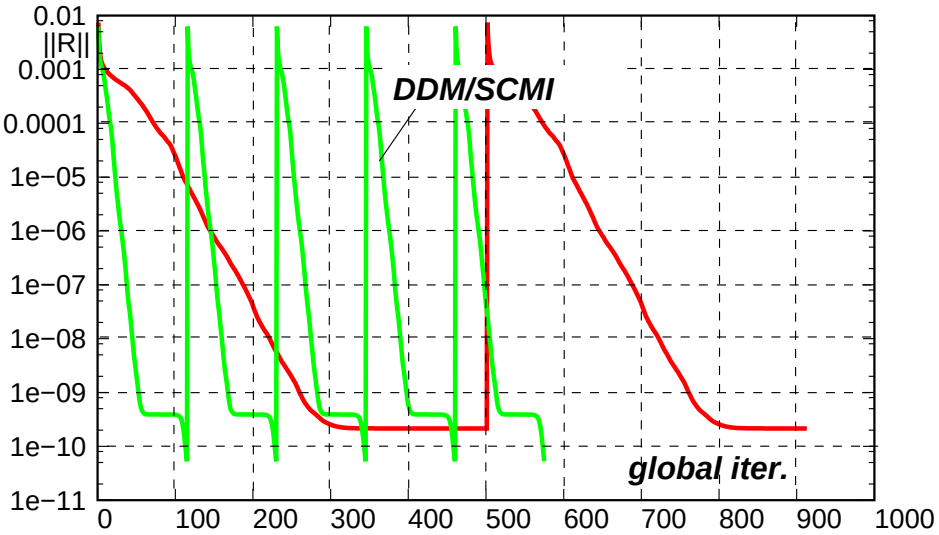

Figura 5: Navier Stokes, cavidad cúbica. 625000 elementos. Tolerancia 10^{-8}

Cuadro 3: Tiempo para la solución del problema elástico, test con 56000 elementos

Estrategia de solución	Precondicionador	Tiempo de solución	Tiempo relativo
GGMRES	Jacobi	64,85 [sec]	3,33
IISD	Jacobi	19,47 [sec]	1,00
IISD	IS	16,93 [sec]	0,87

Cuadro 4: Tiempo para la solución del problema elástico, test con 90000 elementos

Estrategia de solución	Precondicionador	Tiempo de solución	Tiempo relativo
GGMRES	Jacobi	197,02 [sec]	3,83
IISD	Jacobi	51,49 [sec]	1,00
IISD	IS	45,99 [sec]	0,89

que brindó una menor cantidad de iteraciones. A medida que aumenta el ancho de la banda del precondiconador IS sus resultados se acercan al de N-N. Para un ancho de 5 elementos el número de iteraciones de IS se aproxima al de N-N pero con un costo computacional bastante menor la resolución se realiza sobre una franja de menor tamaño que el subdominio.

En la figura 7 se muestran los mismos resultados que en la figura 6 pero en este caso se trata de un problema fuertemente convectivo. Se observa aquí que el desempeño del precondicionador IS es superior al N-N aún para anchos pequeños de la banda.

5. CONCLUSIONES

En este trabajo se han presentado diferentes procedimientos para resolver sistemas de ecuaciones, resultantes de aplicación del método de elementos finitos en ingenierías. Los métodos directos no son útiles para resolver grandes sistemas debido a los requerimientos computacionales. En estos casos se utilizan técnicas iterativas. Aún así en grandes sistemas de ecuaciones éstas no siempre convergen adecuadamente. La cantidad de iteraciones aumenta, al empeorar el condicionamiento de las matrices, y los requerimientos de almacenamiento de datos puede también complicar su solución. Los métodos de descomposición del dominio proporcionan un tratamiento intermedio entre las técnicas iterativas puras y los métodos directos. En los casos resueltos han resultado eficientes.

Los métodos iterativos, tanto para la solución global como para el problema de interfaz, en el caso de descomposición de dominio, precisan precondicionamiento para acelerar su convergencia. Se presentan algunas técnicas para construcción de precondicionadores, mostrando su desempeño en diferentes problemas.

REFERENCIAS

[1] Sterling T.L., Salmon J., Becker D.J. and Savarese D.F., “How to build a Beowulf”, The MIT Press, Cambridge, 1999.

[2] Bjørstad, P.E. and Widlund, O.B., “Iterative Methods for the Solution of Elliptic Problems on Regions Partitioned into Substructures”, SIAM J.Num.Analysis, Vol. 23, pp. 1097-

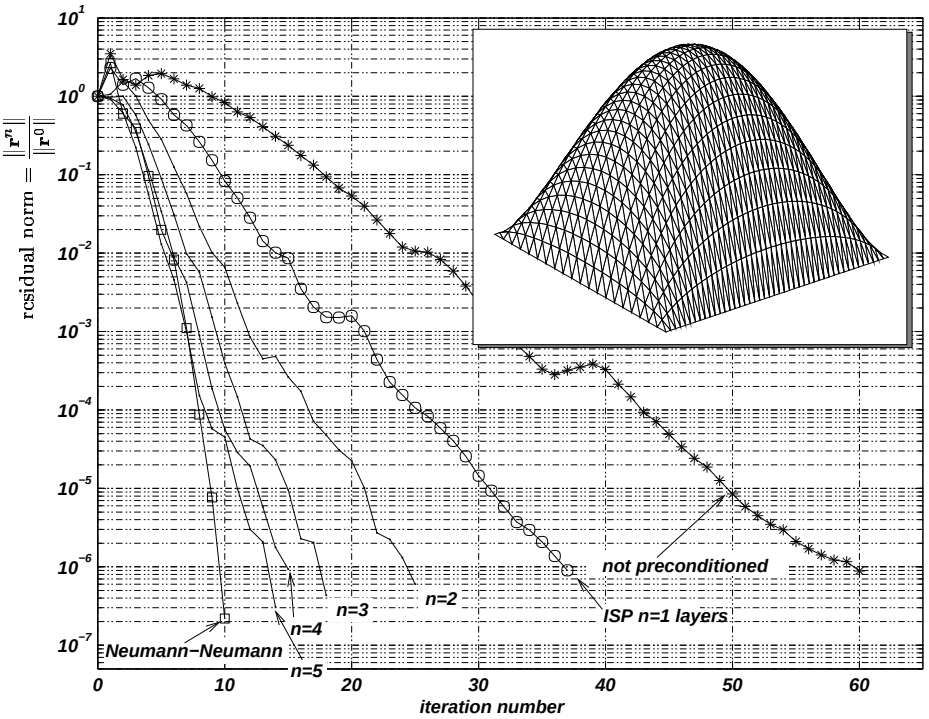

Figura 6: Problema de Poisson

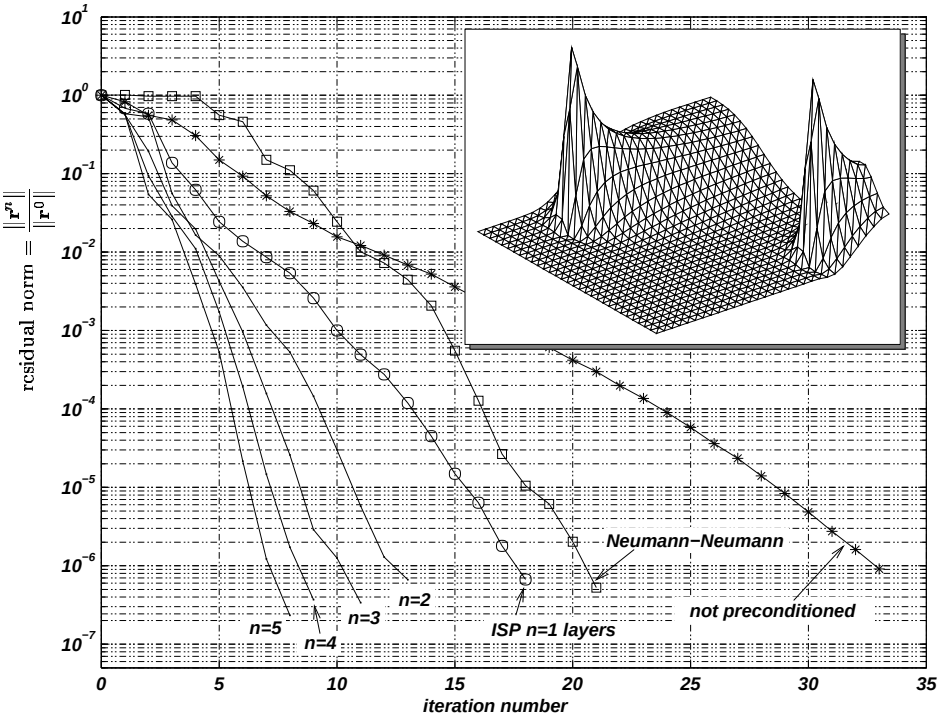

Figura 7: Problema de convección-difusión

1120 (1986).
[3] Bramble, J.H., Pasciak, J.E., Schatz A.H., “The Construction of Preconditioners for Elliptic Problems by Substructuring”, Math. Computations, Vol. 47, pp 103-134 (1986).
[4] Cluster Gerónimo, CIMEC, http://venus.ceride.gov.ar/geronimo.
[5] De Roeck, Y.-H., Le Tallec, P. “Analysis and Test of a Local Domain Decomposition Preconditioned”, in IV Intl.Symp. on Domain Decomposition Meth. for Partial Differential Equations (R. Glowinsky, Y. Kuznetsov, G. Meurant, J. Périaux and O. Widlund, Eds.), SIAM, Philadelphia (1991).
[6] Farhat, Ch., Roux, F.-X., “A Method of Finite Element Tearing and Interconnection and its Parallel Solution Algorithm”, Int.J.Numerical Methods in Engineering, Vol.32, pp 1205-1227 (1991).
[7] Le Tallec, P., De Roeck, Y.-H., Vidrascu, M. “Domain Decomposition Methods for Large Linearly Elliptic 3D Problems”, Tech.Report TR/PA/90/20, Centre Européen de Recherche et de Formation Avancée en Calcul Scientifique, Toulouse, France (1990).
[8] Mandel J., “Balancing Domain Decomposition”, Communications in Numerical Methods in Engineering, Vol. 9, pp 233-241 (1993).
[9] Manteuffel T.A., “The shifted incomplete Cholesky factorization”, Proc. Symp. Sparse Matrix Computation, Knockville, TN, Nov. 1978, SIAM Press (1978).
[10] Nour-Omid, B., “Solving Large linearized Systems in Mechanics”, in Solving Large-Scale Problems in Mechanics (M. Papadrakakis, Ed.), J.Wiley and Sons (1993).
[11] Balay S. et al, “PETSc Web page”, http://www.mcs.anl.gov/petsc (2001).
[12] Sanchez P., Sonzogni V., Huespe A., “Evaluation of a stabilized mixed finite element for solid mechanics problems and its parallel implementation, enviado a Computers and Structures (2004).
[13] Smith B.F., “An Optimal Domain Decomposition Preconditioner for the Finite Element Solution of Linear Elasticity Problems”, SIAM J. Sci.Stat.Comput., Vol 13 (1991).
[14] Sonzogni, V.E., “Domain Decomposition Techniques in Structural Mechanics”, Proc. XV CILAMCE: Congresso Ibero Latino-Americano sobre Metodos Computacionais para Engenharia, pp. 1167-1176. 30 nov- 2 dic, 1994, Belo Horizonte (Brasil) (1994).
[15] Storti M., Dalcín L., Paz R., Yommi A., Sonzogni V., 2002, “An interface strip preconditioner for domain decomposition methods”, Mecánica Computacional Vol. XXI, pp. 2991-3007, (2002).
[16] Storti M., Nigro N., “PETSc-FEM: A General Purpose, Parallel, Multi-Physics FEM Program”, http://minerva.arcride.edu.ar/PETSc-FEM/, on line.
[17] Widlund, O.B., “Iterative Substructuring Methods: Algorithms and Theory for Elliptic Problems in the Plane”, First Int. Symposium on Domain Decomposition Methods for Partial Differential Equations, SIAM, R. Glowinski, G.H. Golub, G.A. Meurant and J. Periaux, eds., Philadelphia, (1988).
[18] Winget J.M. and Hughes T.J.R. , “Solution Algorithms for Nonlinear Transient Heat Conduction Analysis Employing Element-by-Element Iterative Strategies”, Computer Methods in Applied Mechanics and Engineering, 52, 711-815, (1985).