Revista internacional de métodos numéricos para cálculo y diseño en ingeniería, Vol. 2, 1, 27-41 (1986)

PRECONDICIONADORES ELEMENTO-POR-ELEMENTO Y GLOBALES. UNA PERSPECTIVA*.

ARTHUR MULLER**
y
THOMAS J.R. HUGHES***

División of Applied Mechanics
Durand Building
Stanford University
Stanford, CA 94305, USA

SUMARIO

Este trabajo compara una serie de precondicionadores que han atraido la atención de la comunidad científica durante los últimos años. Se resuelven varios problemas estáticos lineales que emanan de discretizaciones por elementos finitos y se comparan el número de iteraciones requerido para convergencia, el tiempo para obtener la solución y aspectos de implementación, para varias técnicas de precondicionamiento. Cuando se analiza una topología bi-dimensional el método de factorización directa todavía parece ser el más efectivo, no sólo debido al menor CPU, sino también porque los requerimientos de almacenamiento son relativamente pequeños. Sin embargo, se concluye que, para topologías tri-dimensionales, la iteración es esencial porque tanto el CPU como el amacenamiento se hacen prohibitivos en el proceso de factorización directa.

SUMMARY

In this paper a series of preconditioners which have atracted the atention of the scientific community in recent years are compared. Several linear estatic problems which arise from finite element discretizations are solved and the number of iterations for convergence, solution time and computer implementation aspects are discussed for seversal preconditioning techniques. It has been found that for three dimentional problems it is essentional to iterate due to both CPU and storage requirements, whereas, for two dimensional problems the factorisation method it is more effective.

INTRODUCCION

El interés creciente en soluciones iterativas durante los últimos años ha emanado del deseo de resolver problemas cada vez más grandes. La referencia (1) provee un buen resumen al día en algoritmos y su implementación. Debido a su simplicidad, el método de Gradiente Conjugado Precondicionado (GCP) ha ganado considerable popularidad,

* Trabajo apoyado por NASA Lewis Research Center bajo contrato n.º NAG 3-319
** Asistente Graduado de Investigación
*** Profesor de Ingeniería Mecánica y Chairman de la División de Mecánica Aplicada.

Recibido: Agosto 1985

© Universidat Politècnica de Catalunya (España)
ISSN 0213-1315

principalmente debido a que el proceso de solución utiliza un número fijo predeterminado de vectores. Se han desarrollado varios precondicionadores, cada uno de los cuales reclama su propio éxito.

En este trabajo se realiza una comparación sistemática de varios de estos precondicionadores para un conjunto de problemas estáticos lineales, que varían desde una simple barra unidimensional a placas de Mindlin, donde el mal condicionamiento juega un papel extremadamente importante. Empleamos los siguientes precondicionadores: precondicionador identidad (CG), precondicionador diagonal (Diag), precondicionador elemento-por-elemento de Crout (EBE), precondicionador elemento-por-elemento de Cholesky (CEBE), Gauss-Seidel simetrizado (SGS), y precondicionadores de tipo desacoplado también conocidos como precondicionadores de Bloque-Sobreextendido-(D/BO). Es importante enfatizar que todos los análisis se efectuaron en la misma máquina un VAX 11-780 y con el mismo programa. Se espera que cualquier deficiencia en la programación haya sido común a todos los procedimientos haciendo así que los resultados sean más significativos.

El trabajo está dividido en cinco partes: el apartado siguiente presenta el algoritmo PCG. Tras ello se ofrece una breve descripción de todos los precondicionadores ya mencionados junto con algunas consideraciones acerca de su implementación. El siguiente apartado describe los problemas analizados junto con los resultados obtenidos y el apartado final presenta las conclusiones de este estudio.

ALGORITMO DE GRADIENTE CONJUGADO PRECONDICIONADO

Consideramos el sistema lineal

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

donde $\mathbf{A}$ es una matriz simétrica definida positiva. El algoritmo PCG se puede describir por medio de los siguientes pasos:

Primer paso. Inicialización:

$$m = 0 \tag{2}$$

$$\mathbf{x}_0 = 0 \tag{3}$$

$$\mathbf{r}_0 = \mathbf{b} \tag{4}$$

$$\mathbf{p}_0 = \mathbf{z}_0 = \mathbf{B}^{-1}\mathbf{r}_0 \tag{5}$$

Segundo paso. Solución corregida y resíduo:

$$\alpha_m = \frac{\mathbf{r} \cdot \mathbf{z}}{\mathbf{p} \cdot \mathbf{A}\mathbf{p}} \tag{6}$$

$$\mathbf{x}_{m+1} = \mathbf{x}_m + \alpha_m \mathbf{p}_m \tag{7}$$

$$\mathbf{r}_{m+1} = \mathbf{r}_m - \alpha_m \mathbf{A}\mathbf{p}_m \tag{8}$$

Tercer paso. Verificación de convergencia:

$$\text{es } \| \mathbf{r}_{m+1} \| < \delta \| \mathbf{r}_o \| ? \tag{9}$$

$$\text{si} \rightarrow \text{detener el proceso} \tag{10}$$

$$\text{si} \rightarrow \text{continúe} \tag{11}$$

Cuarto paso. Dirección conjugada corregida:

$$\mathbf{z}_{m+1} = \mathbf{B}^{-1}\mathbf{r}_{m+1} \tag{12}$$

$$\beta_m = \frac{\mathbf{r}_{m+1} \cdot \mathbf{z}_{m+1}}{\mathbf{r}_m \cdot \mathbf{z}_{m+1}} \tag{13}$$

$$\mathbf{p}_{m+1} = \mathbf{z}_{m+1} + \beta_m \mathbf{p}_m \tag{14}$$

vuelva al segundo paso

Vale la pena mencionar varias características importantes. La primera concierne la matriz precondicionante **B**. Debe notarse que si $\mathbf{B} = \mathbf{A}$ el algoritmo converge en exactamente una iteración. Esto nos lleva a concluir de inmediato que mientras mejor **B** aproxima a **A** más rápida es la convergencia. Evidentemente, el uso de la matriz de rigidez **A** como precondicionador equivale a resolver el sistema (1) por medio de un método directo, que es lo que estamos tratando de evitar. Uno debe también prestar atención al hecho de que el único sistema a resolver implica a **B** como la matriz de coeficientes. La necesidad de emplear un precondicionador que conduzca a un sistema de ecuaciones fácil de resolver es aparente. El valor de δ se fijó a 10^{-3} para los análisis presentados aquí.

Los siguientes aspectos de implementación son también importantes: el algoritmo requiere exactamente cuatro vectores además del precondicionador **B**, estos son: **x**, **r**, **z** y **p**. Evidentemente, **r** puede ocupar al mismo espacio que **b**. También el producto $\mathbf{Ap}_m$ se puede acumular en $\mathbf{z}_m$, siempre que el escalar $\mathbf{r}_m \cdot \mathbf{z}_m$ se guarde para uso futuro cuando se corrige la dirección conjugada $\mathbf{p}_{m+1}$.

Se puede demostrar (2) que el resíduo en el algoritmo GCP decrece de acuerdo con la expresión

$$\| \mathbf{r}_{m+1} \| < \frac{k(\mathbf{B}^{-1}\mathbf{A}) - 1}{k(\mathbf{B}^{-1}\mathbf{A}) + 1} \| \mathbf{r}_m \| \tag{15}$$

donde $k(\cdot)$ representa el número de condicionamiento de su argumento. Por lo tanto, un buen precondicionador es uno que hace el número de condicionamiento del sistema precondicionado $\mathbf{B}^{-1}\mathbf{A}$ lo más cercano a 1 posible.

Otro aspecto implementacional significativo del algoritmo GCP es el procedimiento para calcular el producto **Ap**. En un programa de elementos finitos, se han considerado dos maneras distintas: en la primera –la versión elemento-por-elemento– las matrices

elementales $\mathbf{A}^e$ nunca se ensamblan. En cambio, se almacenan al nivel de elemento y el producto se efectúa elemento-por-elemento, de acuerdo a

$$\mathbf{Ap} = \left(\sum_e \mathbf{A}^e\right) \mathbf{Pp} = \sum_e \mathbf{A}^e\mathbf{p} \tag{16}$$

La segunda manera de evaluar el producto **Ap** implica un esquema de almacenaje disperso *(sparse)*. Aquí la matriz se ensambla pero, para mantener el almacenaje al mínimo, sólo se guardan los elementos distintos de cero en la matriz. Este tipo de esquema necesariamente implica una cierta cantidad de almacenamiento general ya que debe definirse un conjunto de vectores índices para poder localizar los elementos a_{ij} de la matriz. Cuando se utiliza una multiplicación dispersa, debe ponerse especial cuidado para que el proceso de localización no deteriore el algoritmo.

En el analisis presente se prefirió una multiplicación dispersa debido a la reducción en el número de operaciones requeridas para evaluar el producto **Ap** comparado con el esquema elemento-por-elemento. Para describir la estructura de datos empleada, sea n_{eq} el número de ecuaciones en el problema, y m_n el número de ecuaciones conectadas con la ecuación n, con número de ecuación menor o igual a n. Entonces, se puede introducir los siguientes vectores.

Vector	Dimensión
IPOINT	n_{eq}
IADJ	$\sum_{n=1}^{n_{eq}} m_n = M$
SPARSE	$\sum_{n=1}^{n_{eq}} m_n = M$

TABLA 1. Almacenaje para multiplicación no-densa.

IADJ es el vector de adyacencia para la matriz bajo consideración. Para cada ecuación n contiene los números de todas las ecuaciones de número menor o igual a n que están conectadas con n. Porque IADJ es un vector unidimensional, se necesita otro vector de localización para situar el principio del conjunto de adyacencia para cada ecuación. k = IPOINT(n) da la posición de la última ecuación conectada a n, contenida en IADJ(k). El número total de ecuaciones conectadas a n se puede calcular de acuerdo con

$$m_n = \begin{cases} IPOINT(n) - IPOINT(n-1), & \text{if } n \neq 1 \\ IPOINT(1), & \text{if } n = 1 \end{cases} \tag{17}$$

La localización de un elemento a_{ij} en particular se realiza por medio del siguiente procedimiento:

```
MIN = 1
IF(J.NE.1)MIN = IPOINT(J - 1) + 1
MAX = IPOINT(J)
```

```
      DO 10 K = MIN, MAX
      IBAR = IADJ(K)
      IF(IBAR.EQ.I)GOTO20
   10 CONTINUE
   20 AIJ = SPARSE(K)
```

Como ejemplo, consideramos la siguiente matriz:

$$A = \begin{bmatrix} 2 & -1 & 1 & 2 & 0 \\ -1 & 5 & 0 & 0 & -2 \\ 1 & 0 & 3 & 0 & 1 \\ 2 & 0 & 0 & 10 & 0 \\ 0 & -2 & 1 & 0 & 7 \end{bmatrix} \tag{18}$$

Entonces tenemos

$$IPOINT = \begin{bmatrix} 1 & 3 & 5 & 7 & 10 \end{bmatrix} \tag{19}$$

$$IADJ = \begin{bmatrix} 1 & 1 & 2 & 1 & 3 & 1 & 4 & 2 & 3 & 5 \end{bmatrix} \tag{20}$$

$$SPARSE = \begin{bmatrix} 2 & -1 & 5 & 1 & 3 & 2 & 10 & -2 & 1 & 7 \end{bmatrix} \tag{21}$$

PRECONDICIONADORES

Consideramos varios de los precondicionadores de uso más común. Los aspectos de implementación de cada uno se mencionarán en forma breve.

Como veremos, algunos precondicionadores requieren una cantidad extensa de almacenamiento general.

Gradientes Conjugados

Evidentemente, la forma más simple de precondicionador es cuando $\mathbf{B} = \mathbf{I}$ donde $\mathbf{I}$ es la matriz identidad.

Precondicionador Diagonal

La siguiente forma más simple de precondicionador es una multiplicación diagonal como ha propuesto Jennings[3]. Aquí, el precondicionador es una matriz diagonal dada por

$$\mathbf{B} = diag(\mathbf{A}) \tag{22}$$

Este precondicionador tiene dos características importantes: primero, es muy económico ya que el costo se reduce a un producto interior efectuado en coma flotante. Por lo tanto, la vectorización es inmediata. Segundo, tiene un efecto multiplicador favorable que es especialmente importante en casos en que hay presentes variables de estado de dimensiones diferentes.

Precondicionador Elemento-por-Elemento de Crout

Los precondicionadores elemento-por-elemento fueron introducidos por Hughes et al. en un esfuerzo por hacer mejor uso de la estructura de datos provista por los programas de elemento finitos (véase[4] y las referencias citadas allí). Aún cuando existen varios tipos de precondicionadores de elementos, el precondicionador elemento-por-elemento de Crout se ha demostrado más efectivo en la mayoría de los casos. Está definido por

$$\mathbf{B} = \mathbf{W}^{1/2} \times \prod_{e=1}^{N_{el}} L_p\left[\bar{\mathbf{A}}^e\right] \times \prod_{e=1}^{N_{el}} D_p\left[\bar{\mathbf{A}}^e\right] \times \prod_{e=N_{el}}^{1} U_p\left[\bar{\mathbf{A}}^e\right] \times \mathbf{W}^{1/2} \tag{23}$$

donde

$$\bar{\mathbf{A}}^e = \mathbf{I} + \mathbf{W}^{-1/2}\,(\mathbf{A}^e - \mathbf{W}^e)\mathbf{W}^{-1/2} \tag{24}$$

$$\mathbf{W} = diag(\mathbf{A}) \tag{25}$$

$$\mathbf{W}^e = diag(\mathbf{A}^e) \tag{26}$$

$\mathbf{A}^e$ es la i-ésima matriz elemental, y L_p [·], D_p [·], y U_p [·] representan los factores triangular inferior, diagonal y triangular superior respectivamente, en la factorización de Crout.

Debe advertirse que aunque las matrices elementales $\bar{\mathbf{A}}^e$ tienen la misma dimensión que la matriz $\mathbf{A}$, todas las operaciones se restringen al conjunto de ecuaciones conectadas con el elemento bajo consideración. Así, tal como en la formación de la matriz de rigidez y cálculo de esfuerzos, este tipo de precondicionamiento también puede efectuarse al nivel de elemento. Después de que las matrices precondicionadoras elementales se forman y factorizan, la evaluación de $\mathbf{B}^{-1}\mathbf{r}_m$ se hace por medio de un bucle de tres partes sobre todos los elementos, donde se utiliza un procedimiento de "juntar/separar" en la reducción frontal, multiplicación diagonal y substitución hacia atrás en cada elemento. Sin embargo, la experimentación numérica ha demostrado que esto no es tan importante como podría anticiparse[5].

La vectorización de este algoritmo no es directa pero puede realizarse con éxito[6]. Por otro lado, el algoritmo puede paralelizarse fácilmente, una característica importante para aquellos con acceso a procesadores múltiples.

Precondicionador Elemento-por-Elemento de Cholesky

En este caso, el precondicionador está dado por

$$\mathbf{B} = \mathbf{W}^{1/2} \times \prod_{e=1}^{N_{el}} L_p\left[\mathbf{A}^e\right] \times \prod_{e=N_{el}}^{1} U_p\left[\mathbf{A}^e\right] \times \mathbf{W}^{1/2} \tag{27}$$

donde todas las matrices tienen el mismo significado que en el EBE de Crout y L_p [·] y U_p [·] representan respectivamente los factores triangular inferior y superior en la factorización de Cholesky.

La mayor ventaja de Cholesky-EBE sobre Crout-EBE reside en el hecho de que sólo requiere un proceso de juntar/esparcir de dos partes lo que reduce el número de "páginas" en las máquinas virtuales.

Precondicionadores de Bloque Sobreextendido

También conocidos bajo el nombre de precondicionadores desensamblados, estos precondicionadores fueron desarrollados por Hayes y Devloo[7]. El precondicionador desensamblado es similar a un precondicionador elemento-por-elemento. La diferencia principal es que el precondicionador desensamblado implica una **suma** en vez de un **producto**, como es el caso en los precondicionadores elemento-por-elemento de Crout y de Cholesky.

Para describir el precondicionador desensamblado, consideramos la transformación que asocia un elemento A_{ij} en **A** con un elemento $\bar{a}^e_{ij}$. Los elementos $\bar{a}^e_{ij}$ están definidos por

$$\bar{a}^e_{ij} = A_{IJ} \tag{28}$$

donde

$$I = LM(i,e) \tag{29}$$

$$J = LM(j,e) \tag{30}$$

en que i y j son los números de ecuación locales y LM es la matriz de localización que asigna los números de ecuación globales correspondientes. Definimos

$$\mathbf{B}^{-1} = \mathbf{D}^{-1/2}\,\bar{\mathbf{A}}^{-1}\,\mathbf{D}^{-1/2} \tag{31}$$

donde

$$\bar{\mathbf{A}}^{-1} = \mathop{\mathcal{A}}_{e=1}^{N_{el}} (\mathbf{a}^e)^{-1} \tag{32}$$

en que $\mathcal{A}$ es ***el operador*** de ensamblaje de elementos finitos y **D** es la matriz diagonal cuyos elementos son el número de elementos conectados con la ecuación en consideración.

Una vez que los vectores elementales han sido formados y factorizados, el costo por iteración en el precondicionador desensamblado es el mismo que para el precondicionador elemento por-elemento de Crout, excepto que el proceso de "juntar/separar" se realiza sólo una vez por elemento por iteración. El precondicionador desensamblado tiene dos propiedades importantes: se puede paralelizar fácilmente y es independiente del orden de los elementos.

Como ejemplo consideramos el precondicionador de bloque sobreextendido que se obtiene de la discretización por elementos finitos de una barra unidimensional como se muestra en la Figura 1 Para simplificar suponemos que EA = 1.

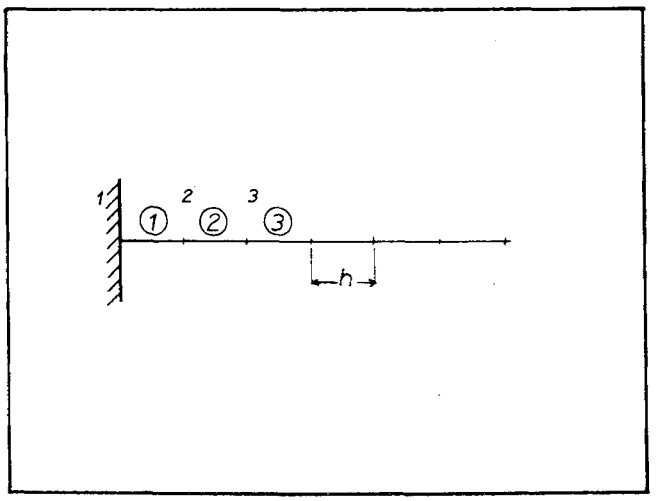

Figura 1

La rigidez de los elementos está dada por

$$\mathbf{a}^e = \frac{1}{h}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{33}$$

donde h es la longitud del elemento. Ensamblando las matrices de rigidez elementales en una matriz golbal **A** obtenemos

$$\mathbf{A} = \frac{1}{h}\begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix} \tag{34}$$

Ahora formamos las matrices $\bar{\mathbf{a}}^e$ "desensamblando" la matriz global **A**:

$$\bar{\mathbf{a}}^1 = \frac{1}{h}[2] \tag{35}$$

$$\bar{\mathbf{a}}^2 = \frac{1}{h}\begin{bmatrix} 2 & -2 \\ -1 & 2 \end{bmatrix} \tag{36}$$

$$\bar{\mathbf{a}}^3 = \frac{1}{h}\begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} \tag{37}$$

Debido a la forma en que se construye el proceso de desensamblaje tenemos la seguridad de que cada $\bar{\mathbf{a}}^e$ está definida positiva. La inversión es entonces directa y nos da

$$(\bar{\mathbf{a}}^1)^{-1} = h[1/2] \tag{38}$$

$$(\bar{\mathbf{a}}^2)^{-1} = h \begin{bmatrix} 2/3 & 1/3 \\ 1/3 & 2/3 \end{bmatrix} \tag{39}$$

$$(\bar{\mathbf{a}}^3)^{-1} = h \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \tag{40}$$

que a su vez

$$\mathop{A}_{e=1}^{N_{el}} (\bar{\mathbf{a}}^e)^{-1} = h \begin{bmatrix} 7/6 & 1/3 & 0 \\ 1/3 & 5/3 & 1 \\ 0 & 1 & 2 \end{bmatrix} \tag{41}$$

La conectividad de los elementos implica

$$\mathbf{D} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{42}$$

Entonces de acuerdo con (31) tenemos

$$\mathbf{B}^{-1} = h \begin{bmatrix} 7/12 & 1/6 & 0 \\ 1/6 & 5/6 & \sqrt{2}/2 \\ 0 & \sqrt{2}/2 & 1 \end{bmatrix} \tag{43}$$

Gauss-Seidel Simetrizado

La utilidad del precondicionador simetrizado de Gauss-Seidel está descrita en forma completa por Nour-Omid[8]. La matriz $\mathbf{A}$ se descompone en la forma

$$\mathbf{A} = \mathbf{L} + \mathbf{D} + \mathbf{L}^T \tag{44}$$

donde $\mathbf{L}$ es una matriz triangular inferior con elementos diagonales cero y $\mathbf{D}$ es la diagonal de $\mathbf{A}$. El precondicionador simetrizado de Gauss-Seidel se define por

$$\mathbf{B} = \frac{1}{2}(2\mathbf{D} + \mathbf{L})\mathbf{D}^{-1}(2\mathbf{D} + \mathbf{L}^T) \tag{45}$$

Nour-Omid[8] ha demostrado que cuando se utiliza el precondicionador simetrizado de Gauss Seidel el algoritmo de gradiente conjugado se puede modificar en tal forma que el tiempo CPU por iteración se puede reducir por un factor cercano a 2. Sin embargo, esta modificación no se utilizó en el análisis que presentamos aquí, de modo que el mismo proceso guía de gradiente conjugado pudiese usarse para todos los problemas.

Debido a la secuencia de las operaciones, el precondiciondor de Gauss-Seiden simetrizado no se puede paralelizar con facilidad en contraste con precondicionadores del tipo elemento por elemento. Las columnas y/o las filas se deben recorrer en un orden predeterminado lo que hace difícil dividir el algoritmo en un conjunto de cálculos independientes que pudiera efectuarse en forma simultánea por un procesador múltiple.

Una ventaja del precondicionador simetrizado de Gauss-Seidel es que prácticamente no requiere almacenamiento adicional si suponemos que se usa un procedimiento de multiplicación dispersa, porque la misma **A** se puede usar para definir **B**.

PROBLEMAS ANALIZADOS

Barra Unidimensional

En el primer problema analizado, una simple barra unidimensional se divide en 100 elementos lineales y se fija en un extremo. Una carga unitaria se aplica en el otro extremo. Se escogieron los siguientes parámetros geométricos y de materiales:

$$L = 1 \text{ (largo)} \tag{46}$$

$$E = 1 \text{ (módulo de Young)} \tag{47}$$

$$A = 1 \text{ (sección de área)} \tag{48}$$

El tiempo en segundos requerido para la solución se representa por t, m es el número de palabras en precisión simple,e i es el número de iteraciones necesarias para obtener convergencia. Todos los análisis se efectuaron en doble precisión. Los resultados se presentan en la Tabla 2.

Método	*t*	*m*	*i*
Directo	0.02	698	–
CG	2.53	1498	100
Diag	2.70	1698	100
EBE	10.03	5910	41
CEBE	8.08	5910	34
D/BO	12.92	6214	65
SGS	2.88	1698	68

TABLA 2. Resultados para el problema de la barra

Viga en Voladizo

Aquí una viga en voladizo se discretiza con 80 elementos de viga de Timoshenko con un punto de quadratura[9], que resulta en 160 ecuaciones. Una carga transversal unitaria se aplica en el extremo libre. Se emplearon las siguientes propiedades geométricas y de materiales:

$$L = 1 \text{ (largo)} \tag{49}$$

$$E = 1 \text{ (módulo de Young)} \tag{50}$$

$$\nu = 0.3 \text{ (cociente de Poisson)} \tag{51}$$

$$h = 0.01 \text{ (espesor)} \tag{52}$$

Los resultados obtenidos se presentan en la Tabla 3.

Método	*t*	*m*	*i*
Directo	0.08	1592	–
CG	*	1498	*
Diag	18.64	1818	324
EBE	39.37	6800	171
CEBE	33.30	6800	144
D/BO	37.93	7044	190
SGS	19.13	3428	213

*No convergió

Tabla 3. Resultados para la viga en voladizo.

Problema de Carga Plana

En este análisis una viga en voladizo se discretiza en una malla de 16 x 8 elementos bilineales, resultando en 288 ecuaciones (véase la Figura 2). Una carga unitaria se aplica en el extremo libre. Se utilizaron las siguientes propiedades geométricas y de materiales:

$$a = 16 \text{ (largo)} \tag{53}$$

$$b = 2 \text{ (espesor)} \tag{54}$$

$$E = 1 \text{ (módulo de Young)} \tag{55}$$

$$\nu = 0.3 \text{ (cociente de Poisson)} \tag{56}$$

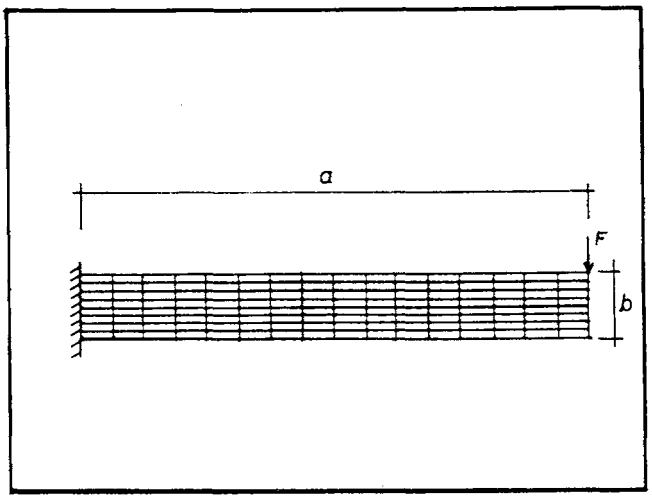

Figura 2

Los resultados obtenidos se presentan en la Tabla 4.

Método	*t*	*m*	*i*
Directo	3.35	19192	–
CG	44.79	9924	282
Diag	37.32	10500	231
EBE	47.37	19732	79
CEBE	42.97	19732	72
D/BO	66.91	20192	132
SGS	38.62	10500	127

Tabla 4. Resultados para el problema de carga plana.

Placa cuadrada

Aquí una placa cuadrada apoyada en los bordes bajo una carga concentrada en el centro se discretiza con una malla de 8 x 8 elementos $T1^{10}$ (debido a la simetría se discretiza sólo un cuarto de la placa) resultando en 736 ecuaciones. Se utilizaron las siguientes propiedades geométricas y de materiales:

$$L = 10 \text{ (largo de los lados)} \tag{57}$$

$$E = 1 \text{ (módulo de Young)} \tag{58}$$

$$t = 0.01 \text{ (espesor)} \tag{59}$$

$$\nu = 0.3 \text{ (coeficiente de Poisson)} \tag{60}$$

Los resultados obtenidos se presentan en la Tabla 5.

Método	*t*	*m*	*i*
Directo	19.95	74562	–
CG	1251.62	34788	2218
Diag	972.39	36260	1717
EBE	2710 38	76220	1195
CEBE	2797.32	76220	1164
D/BO	1082 76	77088	528
SGS	1721 13	36260	1596

Tabla 5. Resultados para el problema de la placa.

Problema de Boussinesq

Para comparar la actuación de los varios precondicionadores en un problema tri-dimensional, consideramos un espacio simi-infinito con una carga unitaria actuando en la superficie libre. Debido a la simetría sólo un cuarto de dominio se discretizó usando una malla de 8 x 8 x 8 elementos tri-lineales, resultando en 1408 ecuaciones. Las propiedades de los materiales son:

$$E = 1 \text{ (módulo de Young)} \tag{61}$$

$$\nu = 0.3 \text{ (coeficiente de Poisson)} \tag{62}$$

Los resultados obtenidos se presentan en la Tabla 6.

Método	*t*	*m*	*i*
Directo	425.64	507728	–
CG	71.16	142824	27
Diag	62.70	145640	23
EBE	162.38	452888	9
CEBE	162.61	452888	9
D/BO	234.81	455076	16
SGS	81.39	145640	14

Tabla 6. Resultados para el problema de Boussinesq.

CONCLUSIONES

De los resultados obtenidos se puede sacar las siguientes conclusiones:

1. El precondicionador diagonal funcionó mejor que el algoritmo de gradiente conjugado sin precondicionamiento. Debe notarse que en los problemas estructurales (viga en voladizo discretizada con elementos de viga y placa apoyada tenemos desplazamientos y rotaciones. El precondicionador diagonal elimina problemas que aparecen debido a las diferentes dimensiones de las variables.

2. La iteración no es siempre preferible. De hecho, en los problemas uni y bi-dimensionales considerados, una solución directa fue más eficiente tanto en términos de CPU como de almacenaje. Sin embargo, debe recalcarse que estos son problemas pequeños. Evidentemente en el caso unidimensional, el método directo será siempre más eficiente que cualquier otro método, debido al ancho de banda extremadamente pequeño.

3. Aunque SGS siempre requirió más iteraciones para converger que EBE, el tiempo total para la solución en SGS fue siempre **menor** que el de EBE. Nour-Omid[8] discute que SGS debería ser siempre más rápido que EBE mientras el número de iteraciones en SGS sea menor que tres veces el de EBE. Debemos recordar también que los tiempos de solución para SGS presentados aquí pueden reducirse casi a la mitad si uno usa el algoritmo descrito en[8]. Sin embargo, tiene la desventaja de que no es adecuado a la paralelización.

4. Los resultados obtenidos con elementos de flexión son interesantes, es decir, los de la viga en voladizo de Timoshenko y la placa de Mindlin. Estos problemas son extremadamente mal condicionados, y no se pueden resolver en forma competitiva usando métodos iterativos a no ser que se utilicen técnicas especiales[11, 12].

5. Las ventajas de los métodos iterativos sobre métodos directos se evidencian en el análisis tri dimensional. Allí se ve que tanto el tiempo de solución como el amacemiento son menores.

6. En estudios previos del problema de Boussinesq[13] el precondicionador elemento-por-elemento resultó mejor que los precondicionadores diagonal y de gradientes conjugados. Se puede concluir de estos resultados que el uso de un esquema con multiplicador disperso es responsable del comportamiento superior de los precondicionadores diagonal y de gradientes conjugados.

7. Se ha demostrado que en problemas con dominios irregulares, los precondicionadores elemento-por-elemento son superiores a los de gradientes conjugados con y sin precondicionador diagonal. De hecho, en algunos problemas de impacto, el precondicionador diagonal no logró converger mientras que el precondicionador elemento-por-elemento logró una convergencia satisfactoria[6].

8. Una conclusión final es que aunque el precondicionador elemento-por-elemento no se comportó tan bien como el precondicionador diagonal, creemos que la solución elemento por elemento se puede todavía mejorar en forma significativa. En esa dirección se han obtenido resultados alentadores que se presentarán en publicaciones futuras.

AGRADECIMIENTOS

Los autores desean agradecer a la Comisión Nacional Brasileña para la Energía Atómica (Comissão Nacional de Energía Nuclear - CNEN) por su ayuda parcial a A. Muller.

REFERENCIAS

1. W.K. Liu, T. Belytschko y K.C. Park, (editors), *Innovative Methods for Nonlinear Problems,* Pineridge Press, Swansea, U.K., (1984).
2. D.G. Luenberger, *Introduction to Linear and Nonlinear Programming,* Addison-Wesley Publishing Company, (1973).
3. A. Jennings y G.M. Malik, "The Solution of Sparse Linear Equations by the Conjugate Gradient Method", *International Journal for Numerical Methods in Engineering,* Vol. **12,** 141-158, (1978).
4. T.J.R. Hughes I. Levit, J. Winget y T. Tezduyar, "New Alternating Direction Procedures in Finite Element Analysis Based Upon EBE Approximate Factorizations", *Computer Methods for Nonlinear Solid and Structural Mechanics,* pp. 75-109, AMD - Vol. **54,** ASME, New York, (1983).
5. J.M. Winget y T.J.R. Hughes, "Solution Algorithms for Nonlinear Transient Heat Conduction Analysis Employing Element-by-Element Iterative Strategies", *Computer Methods in Applied Mechanics and Engineering.* Próxima publicación, (1985).
6. R. Ferencz, Ph.D. thesis, Stanford University (forthcoming)
7. L. Hayes y P. Devloo, "An Element-By-Element Block Iterative Method for Large Nonlinear Problems", *Innovative Methods for Nonlinear Problems,* pp. 51-62, Pineridge Press, Swansea, U.K., (1984).
8. B. Nour-Omid, "A Preconditioned Conjugate Gradient Method for Solution of Finite Element Equations", *Innovative Methods for Nonlinear Problems,* pp. 17-40, Pineridge Press, Swansea, U.K. (1984).
9. T.J.R. Hughes R.L. Taylor y W. Kanoknukulchai, "A Simple and Efficient Finite Element for Plate Bending", *International Journal for Numerical Methods in Engineering,* Vol. **11,** n.º 10, 1543, (1977).
10. T.J.R. Hughes y T.E. Tezduyar, "Finite Elements Based Upon Mindlin Plate Theory With Particular Reference to the Four-Node Bilinear Isoparametric Element", *Journal of Applied Mechanic,* pp. 587-596, Septiembre (1981).
11. A. Muller y T.J.R. Hughes, "Augmented Weak Forms and Element-by-Element Preconditioners: Efficient Iterative Strategies for Structural Finite Elements - A Preliminary Study" , *Research in Structures and Dynamics - 1984,* NASA Conference Publication 2335, pp. 95-109, (1984).
12. A. Muller y T.J.R. Hughes, "Mixed Finite Element Methods and Iterative Solutions: An Algorithm for Structural Finite Element Analysis", *Innovative Methods for Nonlinear Problems,* pp. 1-16, Pineridge Press, Swansea, U.K., (1984).
13. T.J.R. Hughes, A. Raefsky, A. Muller, J. Winget y I. Levit, "A Progress Report on EBE Solution Procedures in Solid Mechanics", *Numerical Methods for Nonlinear Problems,* Vol. **2,** (eds. C. Taylor, E. Hinton, D.R.J. Owen y E. Oñate), pp. 18-26, Pineridge Press, Swansea, U.K., (1984).