Capítulo 4

Método de Wang

4.1 Descripción del método

Dada una matriz tridiagonal A de tamaño $n \times n$

$$A = \begin{bmatrix} a_1 & b_1 & & & \\ c_2 & a_2 & b_2 & & \\ & \ddots & \ddots & \ddots & \\ & & c_{n-1} & a_{n-1} & b_{n-1} \\ & & & c_n & a_n \end{bmatrix}, \tag{4.1}$$

estrictamente diagonal dominante e irreducible, de nuevo se considera el problema general de obtener la solución del sistema

$$A\boldsymbol{x} = \boldsymbol{d}, \tag{4.2}$$

donde

$$\boldsymbol{d} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix}$$

es el vector de términos independientes.

Se supone que existen dos números naturales k y p tales que $k = \dfrac{n}{p}$ y, de nuevo, se considera en A la siguiente partición por bloques

$$A = \begin{bmatrix} A_0 & B_0 & & & \\ C_1 & A_1 & B_1 & & \\ & \ddots & \ddots & \ddots & \\ & & C_{p-2} & A_{p-2} & B_{p-2} \\ & & & C_{p-1} & A_{p-1} \end{bmatrix} \tag{4.3}$$

donde cada uno de los bloques diagonales

$$A_i = \begin{bmatrix} a_{ik+1} & b_{ik+1} & & & \\ c_{ik+2} & a_{ik+2} & b_{ik+2} & & \\ & \ddots & \ddots & \ddots & \\ & & c_{(i+1)k-1} & a_{(i+1)k-1} & b_{(i+1)k-1} \\ & & & c_{(i+1)k} & a_{(i+1)k} \end{bmatrix}, \quad \text{para } i = 0, 1, \ldots, p-1,$$

es una matriz tridiagonal de tamaño $k \times k$ y, para $i = 0, 1, \ldots, p-2$, cada bloque subdiagonal

$$C_{i+1} = \left[\begin{array}{cccc|c} 0 & 0 & \cdots & 0 & c_{(i+1)k+1} \\ \hline & & & & 0 \\ & & O & & \vdots \\ & & & & 0 \\ & & & & 0 \end{array}\right]$$

y superdiagonal

$$B_i = \left[\begin{array}{c|ccc} 0 & & & \\ 0 & & O & \\ \vdots & & & \\ 0 & & & \\ \hline b_{(i+1)k} & 0 & \cdots & 0 \quad 0 \end{array}\right]$$

es una matriz de tamaño $k \times k$ con un sólo elemento no nulo.

En los vectores $\boldsymbol{x}$ y $\boldsymbol{d}$ se considera una partición por bloques acorde con la realizada en la matriz de coeficientes A, es decir

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_0 \\ \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_{p-2} \\ \boldsymbol{x}_{p-1} \end{bmatrix} \quad \text{y} \quad \boldsymbol{d} = \begin{bmatrix} \boldsymbol{d}_0 \\ \boldsymbol{d}_1 \\ \vdots \\ \boldsymbol{d}_{p-2} \\ \boldsymbol{d}_{p-1} \end{bmatrix}, \tag{4.4}$$

con

$$\boldsymbol{x}_i = \begin{bmatrix} x_{ik+1} \\ x_{ik+2} \\ \vdots \\ x_{(i+1)k-1} \\ x_{(i+1)k} \end{bmatrix} \quad \text{y} \quad \boldsymbol{d_i} = \begin{bmatrix} d_{ik+1} \\ d_{ik+2} \\ \vdots \\ d_{(i+1)k-1} \\ d_{(i+1)k} \end{bmatrix}, \quad i = 0, 1, \ldots, p-1.$$

En consecuencia, el sistema (4.2) se puede escribir como

$$\begin{bmatrix} A_0 & B_0 & & & \\ C_1 & A_1 & B_1 & & \\ & \ddots & \ddots & \ddots & \\ & & C_{p-2} & A_{p-2} & B_{p-2} \\ & & & C_{p-1} & A_{p-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_0 \\ \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_{p-2} \\ \boldsymbol{x}_{p-1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{d}_0 \\ \boldsymbol{d}_1 \\ \vdots \\ \boldsymbol{d}_{p-2} \\ \boldsymbol{d}_{p-1} \end{bmatrix}. \tag{4.5}$$

La idea central en que se basa este método consiste en proceder de forma simultánea en los distintos bloques de la partición (4.3) a la eliminación de los elementos situados por encima y por debajo de la diagonal principal de la matriz de coeficientes, realizando las oportunas actualizaciones en los elementos de A afectados en cada paso, hasta transformar A en una matriz diagonal. Al mismo tiempo se realiza una actualización sobre los elementos del vector de términos independientes que resulten afectados por las operaciones sobre la matriz de coeficientes.

El método se puede resumir en los siguientes pasos:

1 Se realiza sobre la matriz A y los vectores $\boldsymbol{x}, \boldsymbol{d}$ la partición (4.5).

2 Se anulan los elementos subdiagonales de los bloques diagonales; este proceso hace que en los bloques subdiagonales aparezcan nuevos elementos no nulos en las columnas ik-ésimas, para $i = 1, 2, \ldots, p-1$, elementos que se denotan por f_j, con $j = k+1, k+2, \ldots, n$.

3 Se anulan los elementos superdiagonales de los bloques diagonales salvo los situados en las columnas ik–ésimas, para $i = 1, 2, \ldots, p-1$, y los elementos no nulos de los bloques superdiagonales; esto hace que en las columnas ik–ésimas, para $i = 1, 2, \ldots, p$, aparezcan nuevos elementos no nulos que se denotan por g_j, con $j = 1, 2, \ldots, n-1$.

4 En cada bloque subdiagonal se anulan los elementos f_j, situados por debajo de la diagonal principal, obteniendo así una matriz triangular superior.

5 La matriz A se diagonaliza anulando los elementos por encima de la diagonal principal y se obtiene la solución de forma inmediata.

A continuación se dará una descripción más detallada del método en general y se irá aplicando al caso particular en que $n = 16$ y $p = 4$, véase la figura 4.1.

Inicialmente en cada bloque A_i, para $i = 0, 1, \ldots, p-1$, se eliminan los elementos situados por debajo de la diagonal principal. Se realizan las siguientes operaciones,

- para $i = 1, 2, \ldots, p-1$,

$$f_{ik+1} \leftarrow c_{ik+1};$$

- para $i = 0, 1, \ldots, p-1$ y $j = 2, 3, \ldots, k$,

$$\left.\begin{array}{rcl} c_{ik+j} & \leftarrow & \frac{c_{ik+j}}{a_{ik+j-1}}, \\ a_{ik+j} & \leftarrow & a_{ik+j} - c_{ik+j} b_{ik+j-1}, \\ d_{ik+j} & \leftarrow & d_{ik+j} - c_{ik+j} d_{ik+j-1}, \end{array}\right\} \tag{4.6}$$

- para $i = 1, 2, \ldots, p-1$ y $j = 2, 3, \ldots, k$,

$$f_{ik+j} \leftarrow -c_{ik+j} f_{ik+j-1}.$$

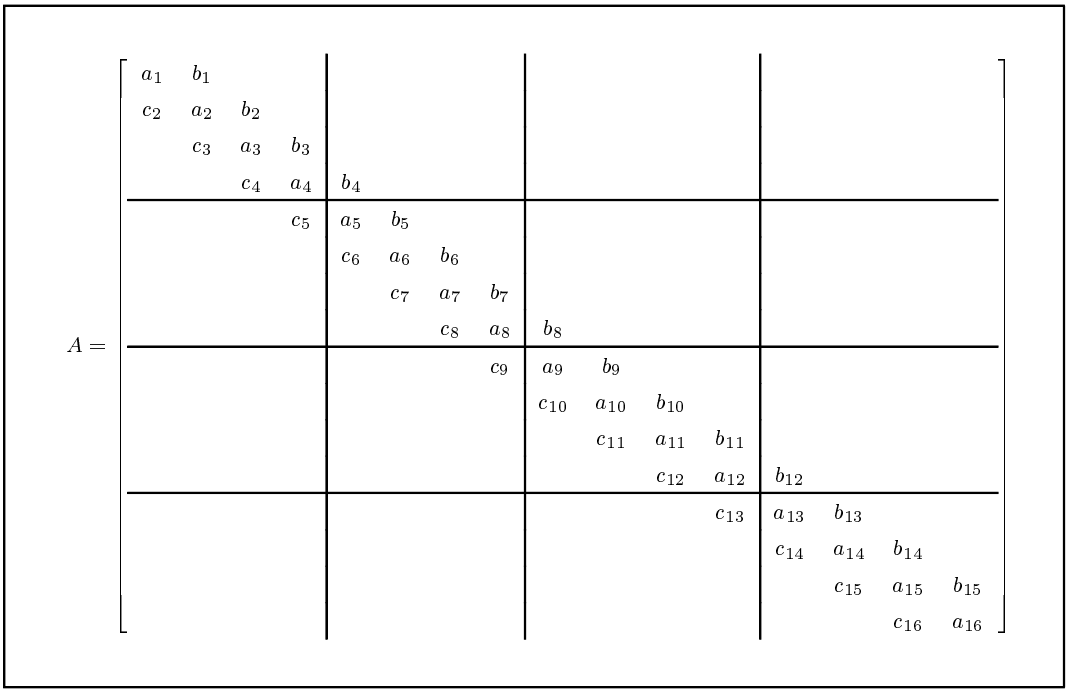


Figura 4.1: *Matriz de coeficientes para el caso* $n = 16$.

Después de este paso inicial, la matriz de coeficientes del caso particular tiene la forma que se muestra en la figura 4.2.

Nótese que en este primer paso aparecen nuevos elementos no nulos f_{ik+j}, para $i = 1, 2, \ldots, p-1$ y $j = 1, 2, \ldots, k$, en las columnas ik-ésimas.

En segundo lugar, en cada bloque diagonal, se anulan los elementos superdiagonales excepto los que se encuentran en las columnas ik-ésimas, para $i = 1, 2, \ldots, p-1$. Se realizan las siguientes operaciones,

- para $i = 0, 1, \ldots, p-1$,

$$g_{(i+1)k-1} \leftarrow b_{(i+1)k-1};$$

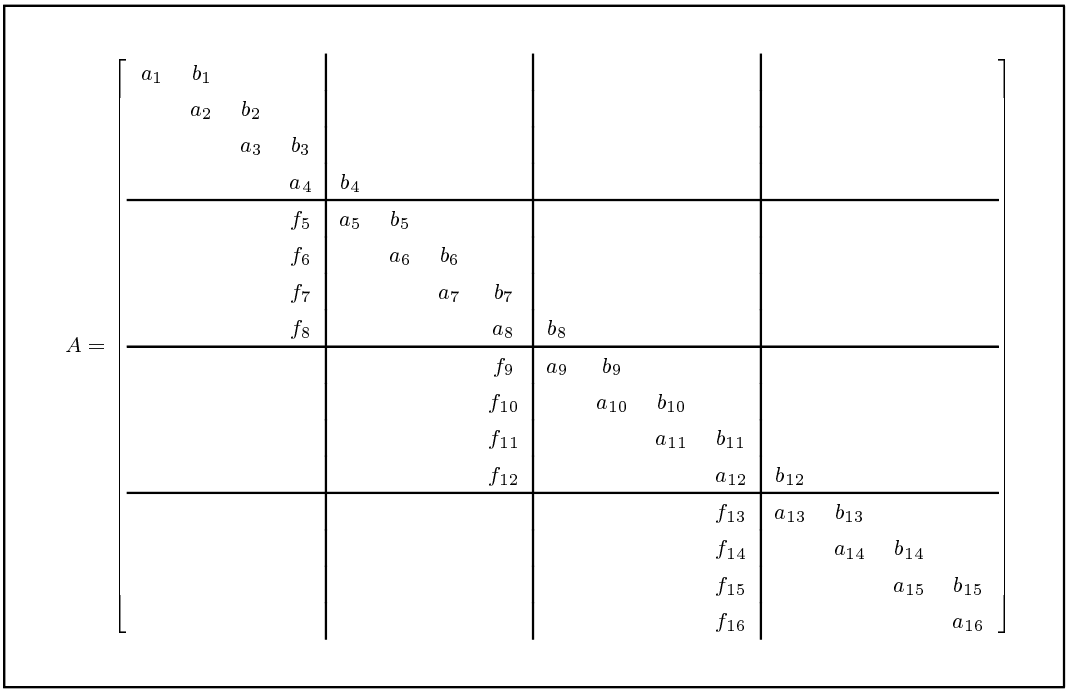


Figura 4.2: *Transformación de la matriz de coeficientes después de la eliminación de los elementos situados por debajo de la diagonal principal, para el caso* $n = 16$.

- para $i = 0, 1, \ldots, p-1$ y $j = 2, 3, \ldots, k-1$,

$$\left.\begin{array}{rcl} b_{(i+1)k-j} & \leftarrow & \frac{b_{(i+1)k-j}}{a_{(i+1)k+1-j}}, \\ g_{(i+1)k-j} & \leftarrow & -b_{(i+1)k-j} g_{(i+1)k+1-j}, \\ d_{(i+1)k-j} & \leftarrow & d_{(i+1)k-j} - b_{(i+1)k-j} d_{(i+1)k+1-j}, \end{array}\right\} \tag{4.7}$$

- para $i = 1, 2, \ldots, p-1$ y $j = 2, 3, \ldots, k-1$,

$$f_{(i+1)k-j} \leftarrow f_{(i+1)k-j} - b_{(i+1)k-j} f_{(i+1)k+1-j}.$$

La figura 4.3 muestra la forma en que queda la matriz tras la realización de estas operaciones.

A continuación, para $i = 1, 2, \ldots, p-1$, se anulan los elementos b_{ik} que se encuentran en las columnas $(ik+1)$-ésimas (en la figura 4.3 son los elementos remarcados). Nótese que para anular estos elementos deben realizarse operaciones elementales en las que intervienen elementos que no se encuentran en la misma fila de la partición (en el sentido de que los

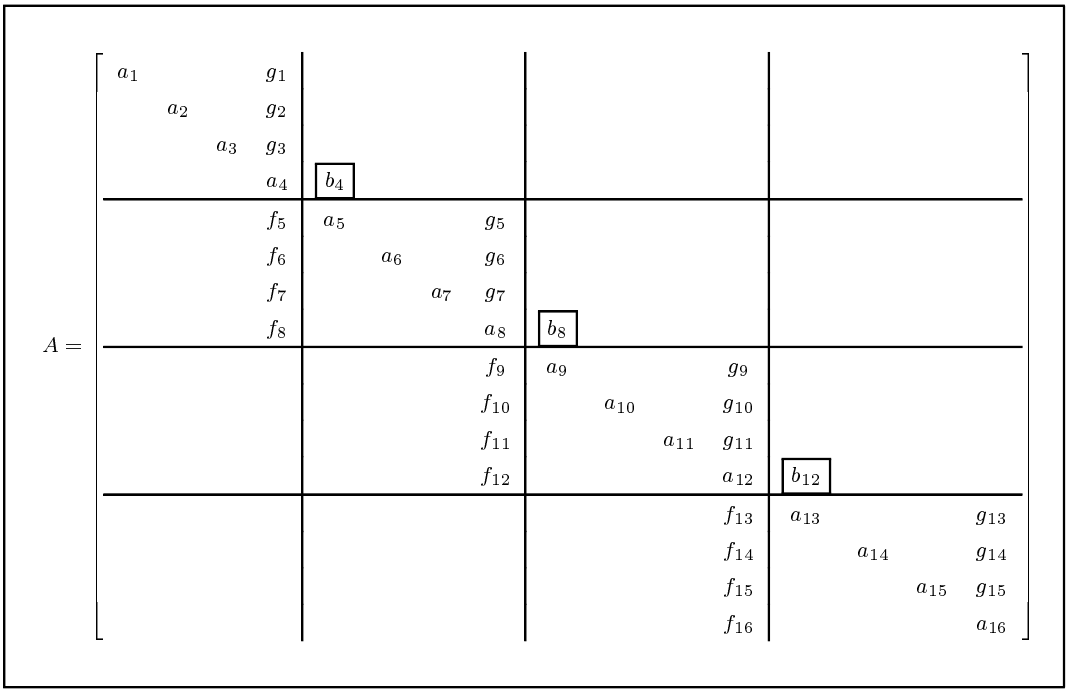


Figura 4.3: *Transformación de la matriz de coeficientes después de la eliminación los elementos situados por encima de la diagonal principal, para el caso* $n = 16$.

subíndices de los bloques C_i, A_i, y B_i coinciden) ya que para eliminar el elemento b_{ik}, que se encuentra en los bloques C_{i-1}, A_{i-1}, B_{i-1}, se efectúan operaciones en las que intervienen elementos de la fila $(ik+1)$-ésima y esta fila forma parte de los bloques C_i, A_i, B_i.

Las operaciones que se llevan a cabo son, para $i = 1, 2, \ldots, p-1$,

$$\left.\begin{array}{rcl} b_{ik} & \leftarrow & \frac{b_{ik}}{a_{ik+1}}, \\ a_{ik} & \leftarrow & a_{ik} - b_{ik} f_{ik+1}, \\ g_{ik} & \leftarrow & -b_{ik} g_{ik+j}, \\ d_{ik} & \leftarrow & d_{ik} - b_{ik} d_{ik+1}. \end{array}\right\} \tag{4.8}$$

Aparecen así nuevos elementos g_{ik} en las columnas $(i+1)k$-ésimas, para $i = 1, 2, \ldots, p-1$.

La figura 4.4 muestra la forma en que queda la matriz del ejemplo una vez anulados los elementos b_4, b_8 y b_{12}; los nuevos elementos g_j han sido remarcados.

Para anular b_{ik}, para $i = 1, 2, \ldots, p-1$, utilizando las expresiones (4.8), se necesi-

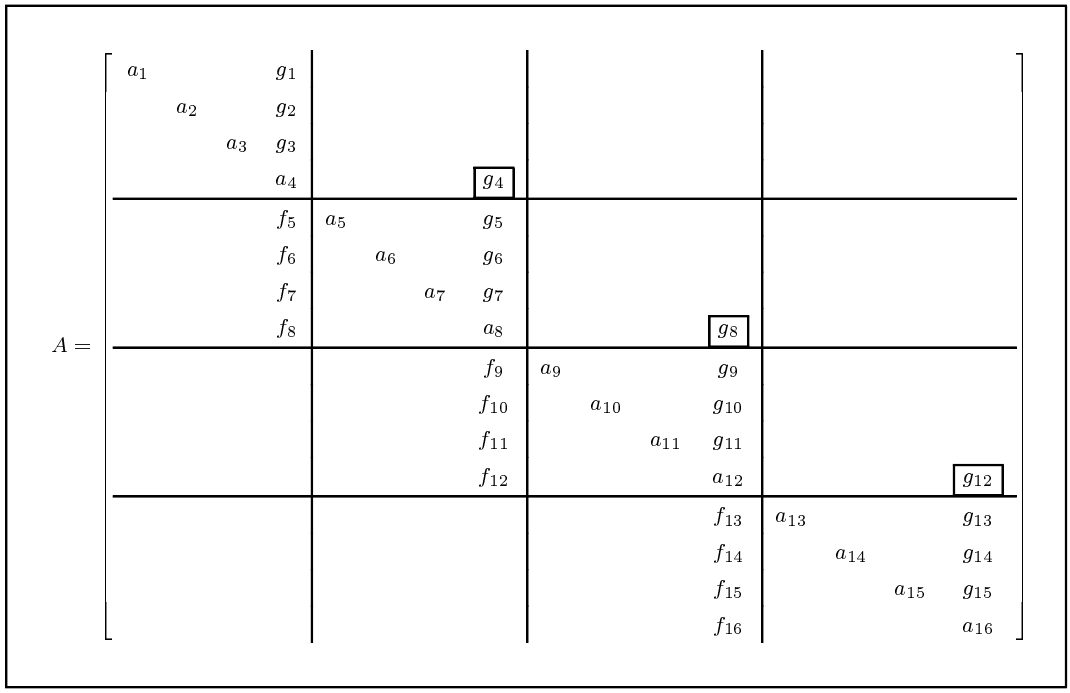


Figura 4.4: *Transformación de la matriz de coeficientes después de anular los elementos b_4, b_8 y b_{12}, para el caso $n = 16$.*

tan elementos que se encuentran en la fila $(ik+1)$-ésima; esto supone una dificultad al paralelizar el método ya que los elementos implicados se encuentran en procesadores distintos, lo que obliga a efectuar una comunicación entre los procesadores para actualizar los elementos en uno de ellos y, una vez actualizados, volverlos a enviar al procesador origen.

El siguiente paso consiste en la eliminación (en cada uno de los bloques diagonales correspondientes) de los elementos f_{ik+j}, para $i = 1, 2, \ldots, p-1$ y $j = 1, 2, \ldots, k$, consiguiendo de este modo que la matriz de coeficientes sea triangular superior. Para eliminar estos elementos se realizan las siguientes operaciones

$$\left.\begin{array}{rcl} f_{ik+j} & \leftarrow & \frac{f_{ik+j}}{a_{ik}}, \\ g_{ik+j} & \leftarrow & g_{ik+j} - f_{ik+j} g_{ik}, \\ d_{ik+j} & \leftarrow & d_{ik+j} - f_{ik+j} d_{ik}. \end{array}\right\}$$

Después de este paso, la matriz del ejemplo queda en la forma que se muestra en la

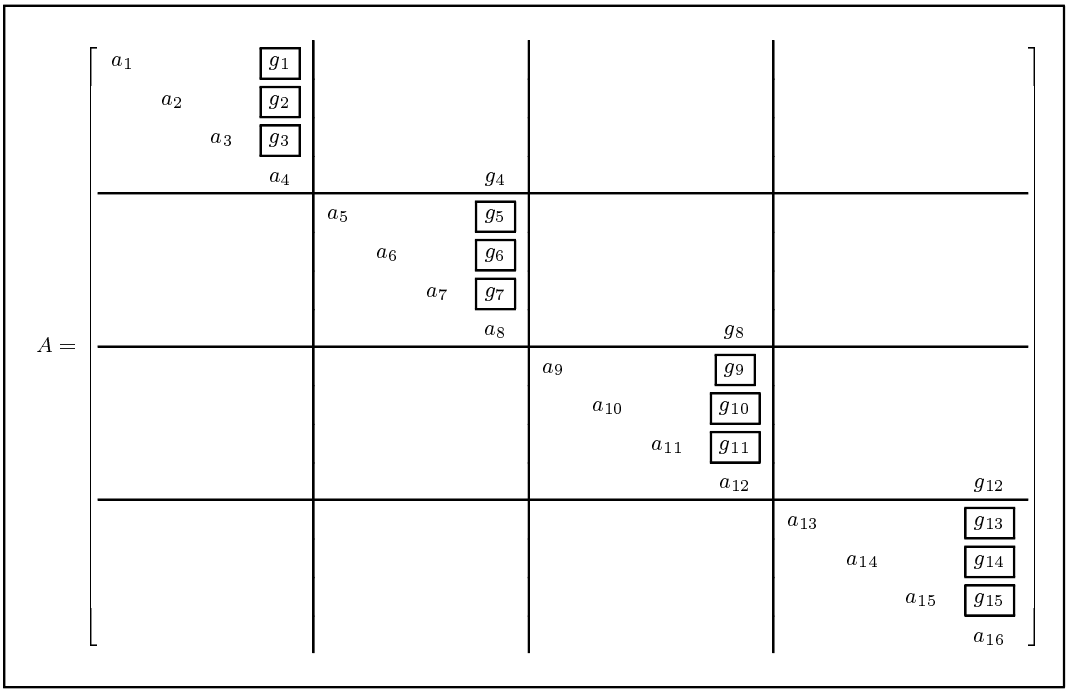


Figura 4.5: *Transformación de la matriz de coeficientes después de anular los elementos* f_{4i+j}*, con* $i = 1, 2, 3$ *y* $j = 1, 2, 3, 4$*, para el caso* $n = 16$*.*

figura 4.5.

Ahora únicamente quedan elementos no nulos fuera de la diagonal principal en las columnas ik-ésimas, con $i = 1, 2, \ldots, p$. Para anularlos se realizan las siguientes operaciones,

- para $i = p-1, p-2, \ldots, 1$ y $j = 1, 2, \ldots, k$,

$$\left.\begin{array}{rcl} g_{ik+j-1} & \leftarrow & \frac{g_{ik+j-1}}{a_{(i+1)\,k}}, \\ d_{ik+j-1} & \leftarrow & d_{ik+j-1} - g_{ik+j-1} d_{(i+1)k}, \end{array}\right\}$$

- para $j = 1, 2, \ldots, k-1$,

$$\left.\begin{array}{rcl} g_j & \leftarrow & \frac{g_j}{a_k}, \\ d_j & \leftarrow & d_j - g_j d_k. \end{array}\right\}$$

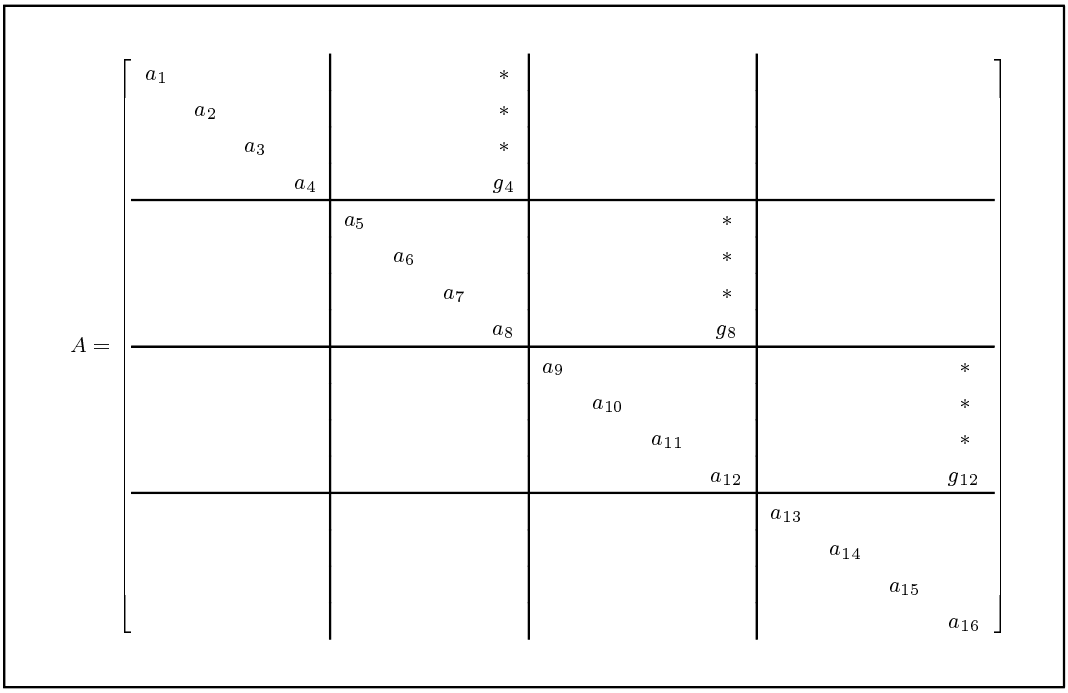


Figura 4.6: *Transformación de la matriz de coeficientes después de anular los elementos* g_{4i+j}*, con* $i = 0, 1, 2, 3$ *y* $j = 1, 2, 3$*, para el caso* $n = 16$*.*

Conviene aclarar que (en principio) el proceso de eliminación de estos elementos no puede efectuarse simultáneamente en todos los bloques diagonales (y por ende en paralelo) ya que al realizar las operaciones elementales necesarias aparecerían nuevos elementos no nulos, por ello el proceso de eliminación comienza con g_{n-1} y finaliza con g_1. La matriz final es diagonal y la solución se obtiene de forma inmediata mediante

$$x_i = \frac{d_i}{a_i}, \quad i = 1, 2, \ldots, n.$$

En las figura 4.6 se muestra cómo quedaría la matriz A del ejemplo si se anulan simultáneamente los elementos g_{4i+j}, con $i = 0, 1, 2, 3$ y $j = 1, 2, 3$ (remarcados en la figura 4.5) y en la figura 4.7 cómo quedaría la matriz A si, a continuación, se eliminan los nuevos elementos no nulos ($*$) que aparecen en la figura 4.6.

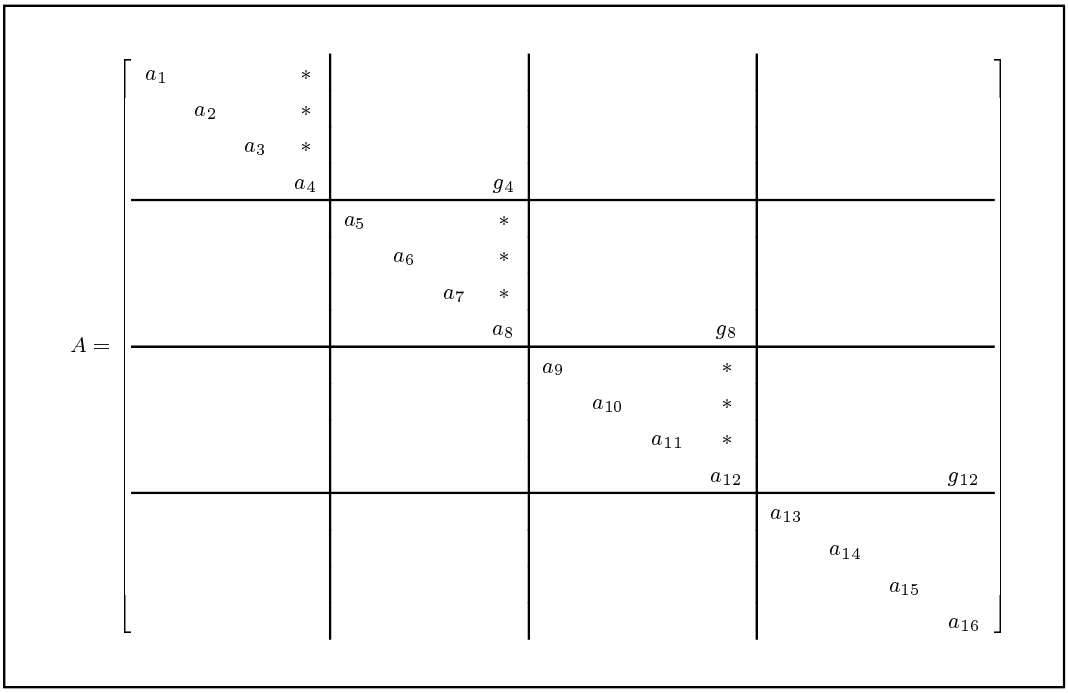


Figura 4.7: *Transformación de la matriz de coeficientes después de eliminar los nuevos elementos no nulos, para el caso* $n = 16$.

4.2 Paralelización del método

Para paralelizar el método se considera que p es el número de procesadores y se asigna a cada uno de los procesadores P_i, para $i = 0, 1, \ldots, p-1$, los bloques C_i, A_i, B_i y $\boldsymbol{d}_i$ de la partición del sistema (4.2) dada en la expresión (4.5). Como ya se ha comentado en la sección 4.1 el método presenta dificultades para su paralelización a la hora de eliminar los elementos interprocesadores b_{ik} que se encuentran en las filas ik-ésimas, para $i = 1, 2, \ldots, p-1$; su anulación obliga a efectuar un paso de comunicación y actualización de los elementos que resultan modificados. Asimismo, la eliminación en paralelo de los elementos g_{ik} (también situados en las filas ik-ésimas), para $i = 1, 2, \ldots, p-1$, presenta una dificultad que será estudiada en esta sección.

El siguiente algoritmo describe la forma de paralelizar el método.

Algoritmo 4.1 Método de las particiones de Wang en paralelo.

Paso 1

Se considera en el sistema (4.2) la partición dada en la expresión (4.5) y se asigna al procesador P_0 los bloques A_0, B_0 y $\boldsymbol{d}_0$, al procesador P_i, para $i = 1, 2, \ldots, p-2$, los bloques C_i, A_i, B_i y $\boldsymbol{d}_i$ y, finalmente, al procesador P_{p-1} los bloques C_{p-1}, A_{p-1}, y $\boldsymbol{d}_{p-1}$.

Paso 2

Cada procesador P_i, para $i = 0, 1, \ldots, p-1$, anula los elementos subdiagonales y superdiagonales (excepto $b_{(i+1)k-1}$) del bloque A_i y realiza las oportunas actualizaciones en el resto de bloques.

Paso 3

Cada procesador P_i, para $i = p-1, p-2, \ldots, 1$, comunica al procesador P_{i-1} los elementos necesarios para que este último anule el elemento b_{ik} y actualice los elementos afectados.

Paso 4

Cada procesador P_{i-1}, para $i = 1, 2, \ldots, p-1$, comunica al procesador P_i los elementos necesarios para que este último anule los elementos f_j, con $j = ik+1, ik+2, \ldots, (i+1)k$, que han aparecido en los bloques C_i y actualice los elementos afectados.

Paso 5

Cada procesador P_i, para $i = 1, 2, \ldots, p-2$ comunica al procesador P_0 los valores actualizados de los bloques A_i, B_i y $\boldsymbol{d}_i$ y el procesador P_{p-1} comunica a P_0 los valores actualizados de los bloques A_{p-1} y $\boldsymbol{d}_{p-1}$. El procesador P_0 anula los elementos superdiagonales g_j, para $j = n-1, n-2, \ldots, 1$, de la matriz A y obtiene la solución del sistema de forma inmediata ya que A es diagonal.

El paso 5 del algoritmo 4.1 puede ser paralelizado tomando los elementos que se encuentran en las filas interprocesadores y formando un sistema auxiliar

$$H\bar{\boldsymbol{x}} = \bar{\boldsymbol{d}}, \tag{4.9}$$

cuya solución, en cada procesador, permite obtener a cada procesador el resto de compo-

nentes de su solución parcial.

En el siguiente ejemplo se va a obtener el sistema auxiliar (4.9) y a describir la forma de obtener la solución parcial, en el caso particular en que $n = 16$ y $p = 4$. A continuación se estudiará el caso general.

Ejemplo 4.1 *Se parte inicialmente de la matriz de la figura (4.4), en la que los únicos elementos que se encuentran fuera de la diagonal principal están en las columnas ik-ésimas, para $i = 1, 2, \ldots, 4$. El sistema auxiliar (4.9) tiene la forma*

$$\begin{bmatrix} a_1 & g_1 & & & \\ f_4 & a_4 & g_4 & & \\ & f_8 & a_8 & g_8 & \\ & & f_{12} & a_{12} & g_{12} \\ & & & f_{16} & a_{16} \end{bmatrix} \begin{bmatrix} x_1 \\ x_4 \\ x_8 \\ x_{12} \\ x_{16} \end{bmatrix} \begin{bmatrix} d_1 \\ d_4 \\ d_8 \\ d_{12} \\ d_{16} \end{bmatrix}. \tag{4.10}$$

Nótese que H es una matriz cuadrada tridiagonal de tamaño $p+1 = 5$ con la característica de que $f_4 = 0$.

La idea de la paralelización del método consiste en que cada procesador reciba los elementos necesarios para formar y resolver el sistema auxiliar (4.9) y así poder seguir trabajando en paralelo para obtener el resto de componentes de su solución parcial.

En el ejemplo, una vez resuelto en cada procesador el sistema auxiliar (4.10), cada procesador conoce las componentes x_1, x_4, x_8, x_{12} y x_{16} y debe proceder a obtener el resto de componentes de su solución parcial. Nótese que cada procesador debe calcular $k - 1$ componentes de su solución parcial a excepción del primero que sólo necesita calcular $k - 2$ componentes, ya que x_1 ha sido obtenida al resolver el sistema auxiliar (4.10). Por comodidad en la notación y puesto que no se aumenta el coste computacional del método, se considera que el procesador P_0 calcula el mismo número de componentes de su solución parcial que el resto.

El procesador P_0 tiene el subsistema

$$\left.\begin{array}{rcrcrcrcl} a_1x_1 & & & + & g_1x_4 & & & = & d_1 \\ & a_2x_2 & & + & g_2x_4 & & & = & d_2 \\ & & a_3x_3 & + & g_3x_4 & & & = & d_3 \\ & & & & a_4x_4 & + & g_4x_8 & = & d_4 \end{array}\right\}$$

y conoce las valores de x_4 y x_8 por lo que únicamente debe calcular x_1, x_2 y x_3, de manera que debe resolver el sistema

$$\left.\begin{array}{rcrcrcl} a_1x_1 & & & + & g_1x_4 & = & d_1 \\ & a_2x_2 & & + & g_2x_4 & = & d_2 \\ & & a_3x_3 & + & g_3x_4 & = & d_3 \end{array}\right\}.$$

Despejando las incógnitas x_1, x_2 y x_3 del sistema anterior se tiene

$$x_i = \frac{1}{a_i}\left(d_i - g_i x_4\right), \quad i = 1, 2, 3.$$

El procesador P_1 tiene el subsistema

$$\left.\begin{array}{rcrcrcrcrcl} f_5x_4 & + & a_5x_5 & & & & + & g_5x_8 & & = & d_5 \\ f_6x_4 & + & & a_6x_6 & & & + & g_6x_8 & & = & d_6 \\ f_7x_4 & + & & & a_7x_7 & + & g_7x_8 & & & = & d_7 \\ f_8x_4 & + & & & & & a_8x_8 & + & g_8x_{12} & = & d_8 \end{array}\right\}$$

y conoce los valores de x_4, x_8 y x_{12} por lo que, únicamente, debe calcular x_5, x_6 y x_7, de manera que debe resolver el sistema

$$\left.\begin{array}{rcrcrcrcl} f_5x_4 & + & a_5x_5 & & & + & g_5x_8 & = & d_5 \\ f_6x_4 & + & & a_6x_6 & & + & g_6x_8 & = & d_6 \\ f_7x_4 & + & & & a_7x_7 & + & g_7x_8 & = & d_7 \end{array}\right\}.$$

Despejando las incógnitas x_5, x_6 y x_7 se obtiene

$$x_i = \frac{1}{a_i}\left(d_i - g_i x_8 - f_i x_4\right), \quad i = 5, 6, 7.$$

Razonando de igual manera, se obtiene en P_2

$$x_i = \frac{1}{a_i}\left(d_i - g_i x_{12} - f_i x_8\right), \quad i = 9, 10, 11,$$

y en P_3

$$x_i = \frac{1}{a_i}\left(d_i - g_i x_{16} - f_i x_{12}\right), \quad i = 13, 14, 15.$$

Mediante esta técnica se consigue que el proceso de cálculo de las soluciones se realice en paralelo hasta el final del mismo. ■

Se puede generalizar el resultado obtenido en el ejemplo 4.1 al caso en que la matriz A es de orden n y se trabaja con p procesadores. Se suponen realizadas en A las operaciones elementales necesarias para que los únicos elementos de A que se encuentren fuera de la diagonal principal estén en las columnas ik-ésimas, para $i = 1, 2, \ldots, p$ (esto es, la matriz A tiene una forma similar a la de la figura (4.4). Ahora, el sistema (4.9) tiene la forma

$$\begin{bmatrix} a_1 & g_1 & & & & & \\ f_k & a_k & g_k & & & & \\ & f_{2k} & a_{2k} & g_{2k} & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & f_{(p-1)k} & a_{(p-1)k} & g_{(p-1)k} & \\ & & & & f_{pk} & a_{pk} & \end{bmatrix} \begin{bmatrix} x_1 \\ x_k \\ x_{2k} \\ \vdots \\ x_{(p-1)k} \\ x_{pk} \end{bmatrix} = \begin{bmatrix} d_1 \\ d_k \\ d_{2k} \\ \vdots \\ d_{(p-1)k} \\ d_{pk} \end{bmatrix}. \tag{4.11}$$

El procesador P_0 tiene el subsistema

$$\left.\begin{array}{rcccccccc} a_1x_1 & & & & + & g_1x_k & & = & d_1 \\ & a_2x_2 & & & + & g_2x_k & & = & d_2 \\ & & \ddots & & \vdots & & & \vdots & \\ & & & a_{k-1}x_{k-1} & + & g_{k-1}x_k & & = & d_{k-1} \\ & & & & & a_kx_k & + \; g_kx_{2k} & = & d_k \end{array}\right\},$$

los valores de x_k y x_{2k} son obtenidos al resolver el sistema auxiliar (4.9) por lo que únicamente debe calcular x_j, para $j = 1, 2, \ldots, k-1$, de manera que debe resolver el

sistema

$$\left.\begin{array}{rrrrrrrrr} a_1x_1 & & & & & + & g_1x_k & = & d_1 \\ & a_2x_2 & & & & + & g_2x_k & = & d_2 \\ & & \ddots & & & \vdots & & \vdots & \\ & & & a_{k-1}x_{k-1} & + & g_{k-1}x_k & = & d_{k-1} \end{array}\right\}.$$

Despejando se tiene

$$x_j = \frac{1}{a_j}\left(d_j - g_j x_k\right), \quad \text{para} \quad j = 1, 2, \ldots, k-1.$$

El procesador P_i, para $i = 1, 2, \ldots, p-2$, tiene el subsistema

$$\left.\begin{array}{rcrcrcrcrcrcl} f_{ik+1}x_{ik} & + & a_{ik+1}x_{ik+1} & & & & & & & + & g_{ik+1}x_{(i+1)k} & = & d_{ik+1} \\ f_{ik+2}x_{ik} & + & & & a_{ik+2}x_{ik+2} & & & & & + & g_{ik+2}x_{(i+1)k} & = & d_{ik+2} \\ & \vdots & & & & & \ddots & & & \vdots & & \vdots & \\ f_{(i+1)k-1}x_{ik} & + & & & & & & & a_{(i+1)k-1}x_{(i+1)k-1} & + & g_{(i+1)k-1}x_{(i+1)k} & = & d_{(i+1)k-1} \\ f_{(i+1)k}x_{ik} & + & & & & & & & a_{(i+1)k}x_{(i+1)k} & + & g_{(i+1)k}x_{(i+2)k} & = & d_{(i+1)k} \end{array}\right\},$$

los valores de x_{ik}, $x_{(i+1)k}$ y $x_{(i+2)k}$ son obtenidos al resolver el sistema auxiliar (4.9) por lo que únicamente debe calcular x_{ik+j}, para $j = 1, 2, \ldots, k-1$, de manera que debe resolver el sistema

$$\left.\begin{array}{rcrcrcrcrcrcl} f_{ik+1}x_{ik} & + & a_{ik+1}x_{ik+1} & & & & & & & + & g_{ik+1}x_{(i+1)k} & = & d_{ik+1} \\ f_{ik+2}x_{ik} & + & & & a_{ik+2}x_{ik+2} & & & & & + & g_{ik+2}x_{(i+1)k} & = & d_{ik+2} \\ & \vdots & & & & & \ddots & & & \vdots & & \vdots & \\ f_{(i+1)k-1}x_{ik} & + & & & & & & & a_{(i+1)k-1}x_{(i+1)k-1} & + & g_{(i+1)k-1}x_{(i+1)k} & = & d_{(i+1)k-1} \end{array}\right\}.$$

Despejando se tiene

$$x_{ik+j} = \frac{1}{a_{ik+j}}\left(d_{ik+j} - g_{ik+j}x_{(i+1)k} - f_{ik+j}x_{ik}\right), \quad \text{para} \quad j = 1, 2, \ldots, k-1.$$

El procesador P_{p-1} tiene el subsistema

$$\left.\begin{array}{rcrcrcrcrcrcl} f_{n-k+1}x_{n-k} & + & a_{n-k+1}x_{n-k+1} & & & & & & & + & g_{n-k+1}x_n & = & d_{n-k+1} \\ f_{n-k+2}x_{n-k} & + & & & a_{n-k+2}x_{n-k+2} & & & & & + & g_{n-k+2}x_n & = & d_{n-k+2} \\ & \vdots & & & & & \ddots & & & \vdots & & \vdots & \\ f_{n-1}x_{n-k} & + & & & & & & & a_{n-1}x_{n-1} & + & g_{n-1}x_n & = & d_{n-1} \\ f_nx_{n-k} & + & & & & & & & & & a_nx_n & = & d_n \end{array}\right\},$$

los valores de x_{n-k} y x_n son obtenidos al resolver el sistema auxiliar (4.9) por lo que únicamente debe calcular x_{n-k+j}, para $j = 1, 2, \ldots, k-1$, de manera que debe resolver el sistema

$$\left.\begin{array}{lcccccccccc}
f_{n-k+1}x_{n-k} & + & a_{n-k+1}x_{n-k+1} & & & & & & + & g_{n-k+1}x_n & = & d_{n-k+1} \\
f_{n-k+2}x_{n-k} & + & & a_{n-k+2}x_{n-k+2} & & & & & + & g_{n-k+2}x_n & = & d_{n-k+2} \\
 & \vdots & & & \ddots & & & & \vdots & & \vdots & \\
f_{n-1}x_{n-k} & + & & & & & a_{n-1}x_{n-1} & + & g_{n-1}x_n & = & d_{n-1}
\end{array}\right\},$$

Despejando se tiene

$$x_{n-k+j} = \frac{1}{a_{n-k+j}}\left(d_{n-k+j} - g_{n-k+j}x_n - f_{n-k+j}x_{n-k}\right), \quad \text{para} \quad j = 1, 2, \ldots, k-1.$$

Si se supone $f_1 = f_2 = \cdots = f_k = 0$, se puede escribir, para $i = 0, 1, \ldots, p-1$,

$$x_{ik+j} = \frac{1}{a_{ik+j}}\left(d_{ik+j} - g_{ik+j}x_{(i+1)k} - f_{ik+j}x_{ik}\right), \quad \text{con} \quad j = 1, 2, \ldots, k-1. \tag{4.12}$$

Nótese que, utilizando (4.12) se puede pasar de la matriz en la forma de la figura (4.4) a la obtención de la solución en cada procesador. Para ello, es necesario un paso de comunicación en el que cada procesador envía a los demás los elementos necesarios para que todos puedan construir el sistema auxiliar (4.9), esta comunicación debe realizarse justo después de la eliminación de los elementos b_{ik}, para $i = 1, 2, \ldots, p-1$, lo que conlleva la actualización de los elementos a_{ik}, d_{ik} y, por otra parte, que se generen los elementos g_{ik}.

4.3 Algoritmos BSP

El siguiente algoritmo BSP recoge las características del método expuestas en la sección 4.2.

Algoritmo 4.2 Un algoritmo BSP para resolución de sistemas tridiagonales basado en el método de Wang.

Superpaso 1

El procesador P_0 envía al procesador P_i, para $i = 1, 2, \ldots, p-1$, los elementos c_{ik+j}, a_{ik+j}, b_{ik+j} y d_{ik+j}, con $j = 1, 2, \ldots, k$, excepto $b_n = 0$.

Superpaso 2

- Para $i = 0, 1, \ldots, p-1$,
  - El procesador P_i anula los elementos c_{ik+j}, para $j = 2, 3, \ldots, k$, utilizando la expresión (4.6).
  - El procesador P_i anula los elementos $b_{(i+1)k-j}$, $j = 2, 3, \ldots, k-1$, utilizando la expresión (4.7).

- Para $i = 1, 2, \ldots, p-1$, el procesador P_{i-1} envía al procesador P_i los elementos a_{ik}, b_{ik} y d_{ik}.

Superpaso 3

- Para $i = 1, 2, \ldots, p-1$,
  - El procesador P_{i-1} anula el elemento b_{ik}.
  - El procesador P_i actualiza los elementos a_{ik}, d_{ik} y calcula el elemento g_{ik} por medio de (4.8).

- El procesador P_0 comunica al resto de procesadores los elementos a_1, g_1 y d_1.
- Para $i = 1, 2, \ldots, p-2$, el procesador P_i comunica al resto de procesadores los elementos a_{ik}, g_{ik}, d_{ik} y $f_{(i+1)k}$.
- El procesador P_{p-1}, comunica al resto de procesadores los elementos $a_{(p-1)k}$, $g_{(p-1)k}$, $d_{(p-1)k}$, f_{pk} y a_{pk}.

Superpaso 4

- Para $i = 0, 1, \ldots, p-1$,
  - P_i construye el sistema auxiliar (4.9), utilizando la expresión (4.11).
  - P_i resuelve el sistema auxiliar (4.9) por el método de Gauss.
  - P_i calcula las componentes de la solución que le corresponden utilizando la expresión (4.12).
- Para $i = 1, 2, \ldots, p-1$, P_i envía a P_0 su solución parcial.

La resolución del sistema auxiliar en cada procesador es una fase fundamental del algoritmo 4.2 ya que permite la obtención de la solución del sistema (4.2) en paralelo. Sin embargo, presenta el inconveniente de que para un número de procesadores alto el orden del sistema auxiliar (4.9) es grande y, además, debe resolverse el mismo sistema en todos los procesadores al mismo tiempo, lo que supone una pérdida de eficiencia en el método. Por esta razón, es interesante encontrar una alternativa a la resolución del sistema (4.9) que permita la eliminación en paralelo de todos los elementos exteriores a la diagonal principal, especialmente en los casos en que el número de procesadores es alto.

Se supone que se han realizado las anulaciones de elementos de la matriz de coeficientes oportunas hasta obtener una matriz triangular superior con forma similar a la de la figura 4.5, esto es, falta aplicar el paso 5 del algoritmo 4.1. Para paralelizar el paso 5, se realiza una modificación sobre el algoritmo 4.1 consistente en la comunicación de cada procesador P_i, para $i = 2, 3, \ldots, p$, a los procesadores P_j, con $j = 1, 2, \ldots, i-1$, de los elementos g_{ik}, $a_{(i+1)k}$ y $d_{(i+1)k}$ y así poder proceder a anular simultáneamente en todos los procesadores los elementos superdiagonales.

El procesador P_i, para $i = 0, 1, \ldots, p-2$, actualiza el elemento $d_{(i+1)k}$ a partir de la relación de recurrencia

$$\left.\begin{array}{rcl} r_p & = & d_n, \\ r_t & = & d_{tk} - \dfrac{g_{tk}}{a_{(t+1)k}} r_{t+1}, \end{array}\right\} \quad \text{para } t = p-1, p-2, \ldots, 2, \tag{4.13}$$

y de los elementos recibidos de los procesadores P_j, para $j = i+1, i+2, \ldots, p-1$, mediante la siguiente operación elemental

$$d_{(i+1)k} \leftarrow d_{(i+1)k} - \frac{g_{(i+1)k}}{a_{(i+2)k}} r_{i+2}. \tag{4.14}$$

Una vez actualizado este valor, elimina los elementos no nulos de cada bloque diagonal y actualiza los correspondientes valores d_{ik+j}, para $j = 1, 2, \ldots, p-1$.

Esta estrategia evita la pérdida de paralelismo en los cálculos finales, conducentes a la diagonalización de la matriz, y ahorra un paso de comunicación al procesador principal.

El algoritmo 4.3 implementa, según el modelo BSP, el método de las particiones de Wang introduciendo la modificación que sugieren las expresiones (4.13) y (4.14).

Algoritmo 4.3 Algoritmo de Wang modificado.

Superpaso 1

Al igual que en el superpaso 1 del algoritmo 4.2, el procesador P_0 envía al procesador P_i, para $i = 1, 2, \ldots, p-1$, los elementos c_{ik+j}, a_{ik+j}, b_{ik+j} y d_{ik+j}, con $j = 1, 2, \ldots, k$, excepto $b_n = 0$.

Superpaso 2

Como en el superpaso 2 del algoritmo 4.2:

- Para $i = 0, 1, \ldots, p-1$,
  - El procesador P_i anula los elementos c_{ik+j}, para $j = 2, 3, \ldots, k$, utilizando (4.6).
  - El procesador P_i anula los elementos $b_{(i+1)k-j}$, para $j = 2, 3, \ldots, k-1$, utilizando (4.7).

- Para $i = 1, 2, \ldots, p-1$, el procesador P_{i-1} envía al procesador P_i los elementos a_{ik}, b_{ik} y d_{ik}.

Superpaso 3

Para $i = 1, 2, \ldots, p-1$,

- El procesador P_{i-1} anula el elemento b_{ik}.
- El procesador P_i actualiza los elementos a_{ik}, d_{ik} y calcula los elementos g_{ik}, por medio de la expresión (4.8).
- El procesador P_i comunica al procesador P_{i-1} los elementos a_{ik}, d_{ik} ya actualizados.

Superpaso 4

Para $i = 1, 2, \ldots, p-1$,

- El procesador P_i anula los elementos f_{ik+j} para $j = 1, 2, \ldots, k$.
- El procesador P_i comunica al procesador P_j, siendo $j < i$, los elementos g_{ik}, $a_{(i+1)k}$ y $d_{(i+1)k}$.

Superpaso 5

- Para $i = 0, 1, \ldots, p-2$,
  - El procesador P_i anula el elemento $g_{(i+1)k}$ y actualiza el elemento $d_{(i+1)k}$ de acuerdo con (4.13) y (4.14).
  - El procesador P_i anula los elementos g_{ik+j} y actualiza los elementos d_{ik+j}, para $j = 1, 2, \ldots, k-1$.
- Para $i = 0, 1, \ldots, p-1$, el procesador P_i calcula

$$x_{ik+j} = \frac{d_{ik+j}}{a_{ik+j}}, \quad \text{con} \quad j = 1, 2, \ldots, k.$$

- Para $i = 1, 2, \ldots, p-1$, el procesador P_i envía su solución parcial al procesador principal P_0.

4.4 Coste computacional de los algoritmos basados en el método de Wang

El coste computacional del algoritmo 4.2 se obtiene sumando los costes individuales de cada uno de sus superpasos.

Coste del superpaso 1. En este superpaso sólo existe comunicación de elementos desde el procesador principal al resto, cada procesador recibe $4k$ elementos de P_0 por lo que el número de elementos comunicados es $4k(p-1)-1$. En consecuencia, el coste del superpaso es

$$(4n-4k-1)g+l. \tag{4.15}$$

Coste del superpaso 2. En cada procesador P_i, para $i=0,1,\ldots,p-1$, el número de operaciones necesarias para anular los elementos c_{ik+j} y actualizar los elementos d_{ik+j}, a_{ik+j} y f_{ik+j}, con $j=2,3,\ldots,k$, es $6(k-1)$; para anular los elementos $b_{(i+1)k-j}$ y actualizar los elementos $g_{(i+1)k-j}$, $d_{(i+1)k-j}$ y $f_{(i+1)k-j}$, con $j=2,3,\ldots,k-1$, se requieren $6(k-2)$ operaciones (véanse las expresiones (4.6) y (4.7)). En consecuencia, el coste aritmético del superpaso es $12k-18$.

Cada procesador P_i, para $i=1,2,\ldots,p-1$, recibe tres elementos del procesador P_{i-1}, lo que supone un coste de comunicación $3g$. El coste total del superpaso es por tanto

$$12k-18+3g+l. \tag{4.16}$$

Coste del superpaso 3. La eliminación de los elementos b_{ik}, para $i=1,2,\ldots,p-1$, supone un total de 6 operaciones en cada uno de los procesadores activos, que corresponden a la actualización de los elementos a_{ik}, d_{ik} y g_{ik}.

En cuanto al coste de comunicación, el procesador principal comunica 3 elementos a los demás, los procesadores centrales comunican al resto 4 elementos y el último comunica a los otros procesadores un total de 5 elementos. Por tanto, el procesador P_{p-1} es el que más elementos envía o recibe ya que envía $5(p-1)$ elementos y recibe $3+4(p-2)$ elementos. En consecuencia, el coste del superpaso 3 es

$$6+5(p-1)g+l. \tag{4.17}$$

Coste del superpaso 4. La formación del sistema auxiliar (4.9) en cada procesador no precisa ninguna operación, su resolución por el método de Gauss necesita un total de $8(p+1)-7$ operaciones, ya que el número de ecuaciones es $p+1$, y el cálculo de cada una de las componentes restantes de la solución parcial, utilizando la expresión (4.12), requiere un total de 6 operaciones por cada una de las $k-1$ componentes. De esta forma, el coste aritmético del superpaso es $8(p+1)-7+6(k-1)=8p+6k-5$.

En este superpaso, cada procesador comunica su solución parcial a P_0, por lo que el procesador principal recibe un total de $n-k$ elementos. Se tiene que el coste del superpaso 4 es

$$6k+8p-5+(n-k)g+l. \tag{4.18}$$

Sumando las expresiones (4.15), (4.16), (4.17) y (4.18) se obtiene que el coste computacional del algoritmo 4.2 es

$$18k+8p-17+(5n-5k+5p-3)g+4l.$$

A continuación se obtiene el coste computacional del algoritmo 4.3.

Coste del superpaso 1. El coste computacional de este superpaso es el mismo que el del superpaso 1 del algoritmo 4.2, que está dado por la expresión (4.15).

Coste del superpaso 2. El coste es el mismo que el del superpaso 2 del algoritmo 4.2, que está dado por la expresión (4.16).

Coste del superpaso 3. En este superpaso, para $i=1,2,\ldots,p-1$, el procesador P_{i-1} anula el elemento b_{ik} y el procesador P_i actualiza los elementos a_{ik}, d_{ik}, g_{ik} para lo que necesita realizar 6 operaciones. Además, P_i comunica los elementos a_{ik}, d_{ik}, al procesador P_{i-1}, lo que supone un coste de comunicación $2g$

El coste total del superpaso 3 es por tanto

$$6+2g+l. \tag{4.19}$$

Coste del superpaso 4. Para eliminar en cada procesador P_i, para $i=1,2,\ldots,p-1$, los elementos f_{ik+j}, con $j=1,2,\ldots,k$, se necesitan $5k$ operaciones aritméticas. El

coste de comunicación de los elementos g_{ik}, $a_{(i+1)k}$ y $d_{(i+1)k}$ desde el procesador P_i, para $i = 1, 2, \ldots, p-1$, a los procesadores P_j, siendo $j < i$, es $3(p-1)g$ ya que el procesador que más elementos recibe es P_0 (recibe $3(p-1)$ elementos) y el procesador que más envía es P_{p-1} (envía $3(p-1)$ elementos). En resumen, el coste del superpaso 4 es

$$5k + (3p-3)g + l. \tag{4.20}$$

Coste del superpaso 5. En este superpaso se distinguen varias fases en el cálculo aritmético. En primer lugar, y en todas las fases para $i = 0, 1, \ldots, p-2$, se anulan los elementos $g_{(i+1)k}$ lo que conlleva la actualización de los elementos $d_{(i+1)k}$; el procesador que más operaciones realiza es P_0 pues debe efectuar $p-1$ divisiones, $p-1$ productos y $p-1$ sumas para actualizar el elemento d_k. En segundo lugar, la anulación de los elementos g_{ik+j}, con $j = 1, 2, \ldots, k-1$, supone $3(k-1)$ operaciones. Finalmente, el cálculo de las componentes x_{ik+j}, con $j = 1, 2, \ldots, k$, de la solución parcial en cada procesador requiere k divisiones. En consecuencia, el coste aritmético total es $3(p-1)+3(k-1)+k = 4k+3p-6$ operaciones.

Con respecto al coste de comunicación, cada procesador envía al procesador principal su vector de soluciones parciales, esto es $k(p-1)$ elementos.

El coste total del superpaso 5 es por tanto

$$(4k + 3p - 6) + (n-k)g + l. \tag{4.21}$$

Sumando las expresiones (4.15), (4.16), (4.19), (4.20) y (4.21) se obtiene que el coste total del algoritmo es

$$21k + 3p - 18 + (5n - 5k + 3p + 1)g + 5l.$$

Nótese que la implementación del algoritmo 4.2 requiere un superpaso menos que el del algoritmo 4.3, que el coste de comunicación del algoritmo 4.2 es mayor que el del algoritmo 4.3 para $p > 2$ y que para p alto y k no muy grande, el coste aritmético del primer algoritmo es mayor que en el algoritmo 4.3 (por ejemplo, para $p = 256$ y $n = 65536$ el algoritmo 4.2 realiza 513 operaciones más que el segundo algoritmo).

s	p	l	g	$n_{\frac{1}{2}}$
45	1	423	2.3	26
	2	3294	9.5	25
	4	5366	12.4	25
	6	8164	12.5	25

(a) *IBM SP2 switch*

s	p	l	g	$n_{\frac{1}{2}}$
45	1	423	2.3	8
	2	20235	709.7	3
	4	54163	1362.6	9
	6	121958	3211.2	9

(b) *IBM SP2 ethernet*

s	p	l	g	$n_{\frac{1}{2}}$
16.4	1	23	0.2	22
	2	2556	6.9	5
	4	5152	7.4	4
	6	7538	6.8	4

(c) *Cluster de PC's*

Tabla 4.1: *Valores de parámetros BSP.*

4.5 Resultados numéricos

En esta sección se analizan los tiempos previstos teóricamente y los tiempos experimentales de los algoritmos 4.2 y 4.3 (a los que se referenciará en las tablas y figuras como WANG1 y WANG2 respectivamente). Las pruebas experimentales se han realizado en el IBM SP2 y el *cluster* de PC's cuyas características se han descrito en la subsección 1.5.3. Por comodidad, en la tabla 4.1 se repiten los parámetros obtenidos para esas máquinas que se muestran en la tabla 1.1.

El tiempo teórico se ha obtenido considerando que el tamaño de bloque de los mensajes que se comunican entre los procesadores es $k = \frac{n}{p}$ y que el coste de comunicación de una

palabra de 32 bits es

$$g(k) = \left(\frac{n_{\frac{1}{2}}}{k} + 1 \right) g_\infty.$$

Los algoritmos 4.2 (WANG1) y 4.3 (WANG2) han sido implementados en Fortran usando la versión v1.3 de la librería BSPLib, se ha generando el sistema (4.2) obteniendo aleatoriamente los elementos de la matriz de coeficientes A y, para simplificar, eligiendo el vector de términos independientes $\boldsymbol{d}$ de manera que la solución sea $\boldsymbol{x} = [1, 1, \ldots, 1]^T$.

En las tablas 4.2, 4.3, 4.4 y figuras 4.8, 4.9, 4.10 se muestran los tiempos teóricos y experimentales, medidos en segundos, de los algoritmos 4.2 (WANG1) y 4.3 (WANG2) en el IBM SP2 utilizando *switch*.

En las tablas 4.5, 4.6, 4.7 y figuras 4.11, 4.12, 4.13 se muestran los tiempos teóricos y experimentales, medidos en segundos, de los algoritmos 4.2 (WANG1) y 4.3 (WANG1) en el IBM SP2 utilizando *ethernet*.

En las tablas 4.8, 4.9, 4.10 y figuras 4.14, 4.15, 4.16 se muestran los tiempos teóricos y experimentales, medidos en segundos, de los algoritmos 4.2 (WANG1) y 4.3 (WANG2) en el *cluster* de PC's.

Los tiempos han sido obtenidos para diferentes tamaños de la matriz de coeficientes, en el IBM SP2 el tamaño varía desde 128 a 524288 para 2 y 4 procesadores y desde 126 a 516096 para 6 procesadores; en el *cluster* de PC's el tamaño de la matriz de coeficientes varía desde 128 a 65536 para 2 y 4 procesadores y desde 126 a 64512 para 6 procesadores.

En las figuras 4.17, 4.18 y 4.19 se muestra el porcentaje de desviación del tiempo experimental con respecto al teórico en función del tamaño n del sistema [1].

Las mayores diferencias entre el tiempo previsto y el experimental se obtienen en los sistemas de tamaño pequeño en los que, por lo general, suele ser mayor el tiempo experimental que el teórico. A medida que aumenta el tamaño de sistema el tiempo obtenido experimentalmente se ajusta mejor al esperado, para tamaños grandes la desviación (salvo alguna excepción) oscila entre el 0% y el 5%.

[1] Se considera que si el tiempo teórico es 100 y el tiempo experimental es 90, se ha producido un 10% de desviación y en cambio si el tiempo experimental es 110 se ha producido un -10% de desviación.

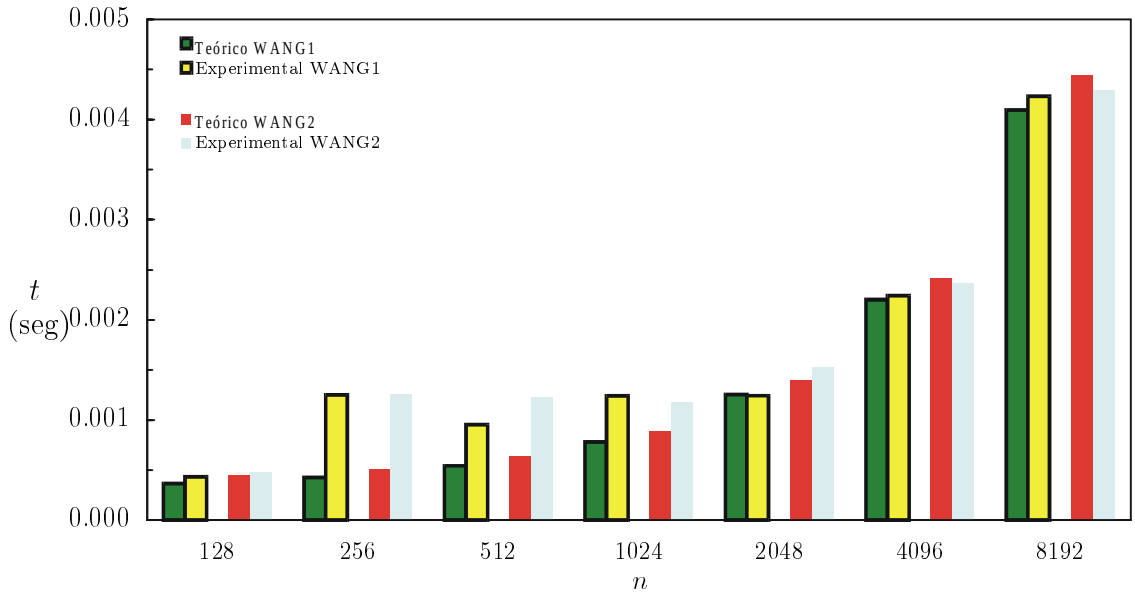
Para las máquinas en las que se han obtenido resultados experimentales, el algoritmo 4.2 (WANG1) es siempre más rápido que el algoritmo 4.3 (WANG2). Esto no sucede cuando el número de procesadores es grande, por ejemplo en un CRAY T3D para $p = 256$ es más rápido el algoritmo 4.3 (WANG2) hasta tamaños de sistema de unas 65000 ecuaciones, como se muestra en la tabla 4.11. Los tiempos se han obtenido en base a los parámeros que se muestran en la tabla 4.13(a)

En la tabla 4.12 se muestra el *speed-up* y la eficiencia de los algoritmos 4.2 (WANG1) y 4.3 (WANG2) con respecto al método de eliminación de Gauss para sistemas tridiagonales, que es el mejor algoritmo secuencial para la resolución de sistemas tridiagonales, considerando los tiempos teóricos de los sistemas de tamaño máximo de los que se han obtenido resultados experimentales en cada una de las máquinas.

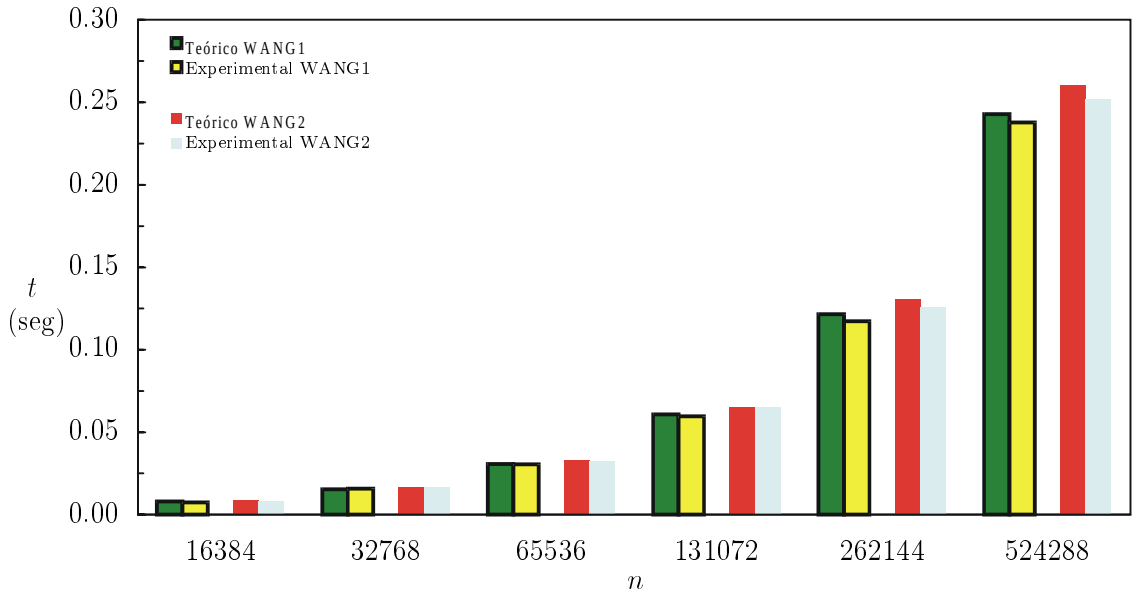
La eficiencia que se obtiene no es muy buena, y en caso del IBM SP2 con *ethernet* es pésima; sin duda un factor determinante es el valor de g. Para otras máquinas con menor valor de g se obtiene mejor eficiencia, como es el caso de un CRAY T3D o un CRAY T3E. Por comodidad, en la tabla 4.13 se repiten los valores de los parámetros de estas máquinas que se muestran en la tabla 2.16, para las mismas se obtiene el *speed-up* y la eficiencia que figura en la tabla 4.14 considerando el mismo tamaño de sistema que para la tabla 4.12.

IBM SP2 2 procesadores *switch*				
	WANG1		WANG2	
n	Teórico	Experimental	Teórico	Experimental
128	0.0004	0.0004	0.0004	0.0005
256	0.0004	0.0013	0.0005	0.0013
512	0.0005	0.0010	0.0006	0.0012
1024	0.0008	0.0012	0.0009	0.0012
2048	0.0013	0.0012	0.0014	0.0015
4096	0.0022	0.0022	0.0024	0.0024
8192	0.0041	0.0042	0.0044	0.0043
16384	0.0079	0.0076	0.0085	0.0083
32768	0.0155	0.0157	0.0166	0.0165
65536	0.0306	0.0305	0.0329	0.0323
131072	0.0609	0.0598	0.0654	0.0654
262144	0.1215	0.1172	0.1304	0.1257
524288	0.2428	0.2379	0.2603	0.2520

Tabla 4.2: *Tiempos teóricos y experimentales de los algoritmos 4.2 (WANG1) y 4.3 (WANG2), medidos en un IBM SP2 con 2 procesadores interconectados mediante switch, para* $128 \leq n \leq 524288$.



(a) $128 \leq n \leq 8192$

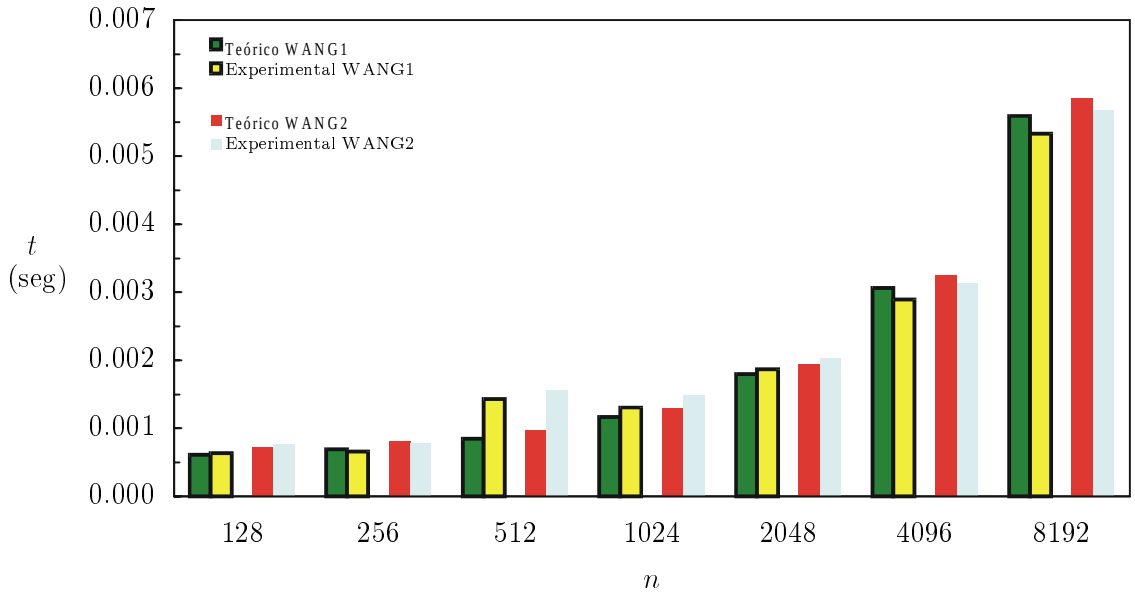


(b) $16384 \leq n \leq 524288$

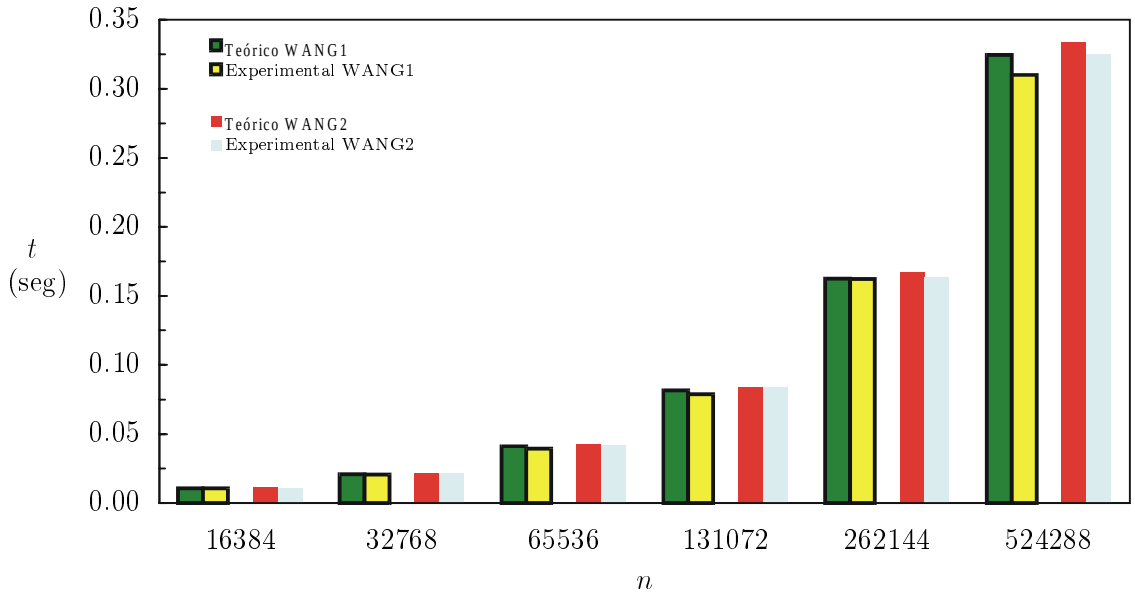
Figura 4.8: *Tiempos teóricos y experimentales de los algoritmos 4.2 (WANG1) y 4.3 (WANG2) en un IBM SP2 con 2 procesadores interconectados mediante switch.*

IBM SP2 4 procesadores *switch*				
	WANG1		WANG2	
n	Teórico	Experimental	Teórico	Experimental
128	0.0006	0.0006	0.0007	0.0008
256	0.0007	0.0007	0.0008	0.0008
512	0.0008	0.0014	0.0010	0.0016
1024	0.0012	0.0013	0.0013	0.0015
2048	0.0018	0.0019	0.0019	0.0020
4096	0.0031	0.0029	0.0032	0.0031
8192	0.0056	0.0053	0.0058	0.0057
16384	0.0107	0.0105	0.0110	0.0107
32768	0.0208	0.0205	0.0214	0.0213
65536	0.0410	0.0394	0.0422	0.0419
131072	0.0815	0.0787	0.0838	0.0837
262144	0.1625	0.1623	0.1670	0.1636
524288	0.3245	0.3102	0.3334	0.3250

Tabla 4.3: *Tiempos teóricos y experimentales de los algoritmos 4.2 (WANG1) y 4.3 (WANG2), medidos en un IBM SP2 con* 4 *procesadores interconectados mediante switch, para* $128 \leq n \leq 524288$.



(a) $128 \leq n \leq 8192$

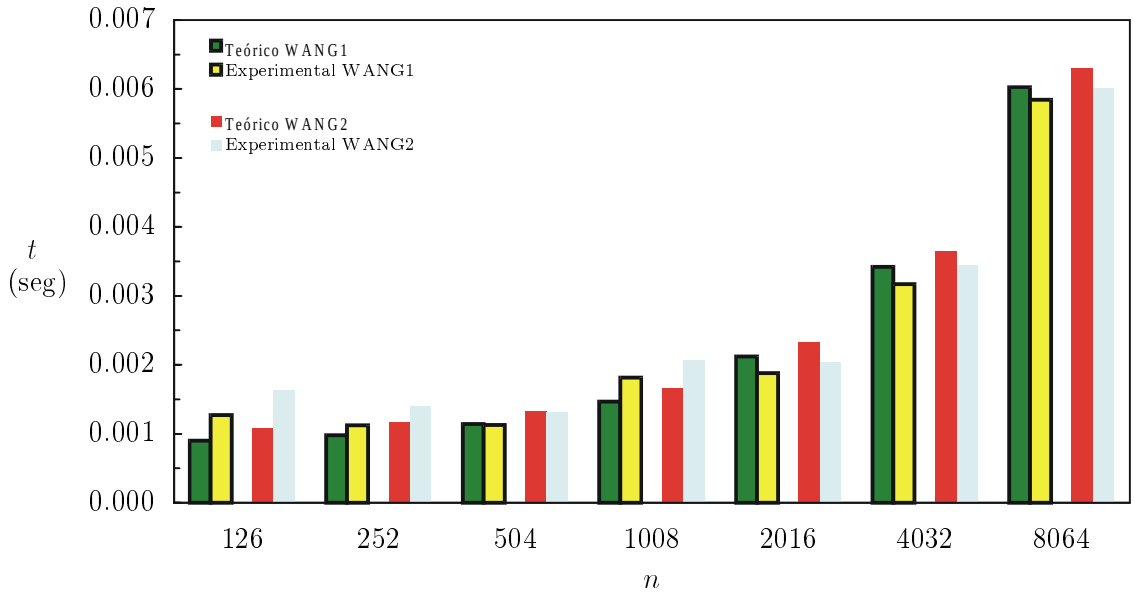


(b) $16384 \leq n \leq 524288$

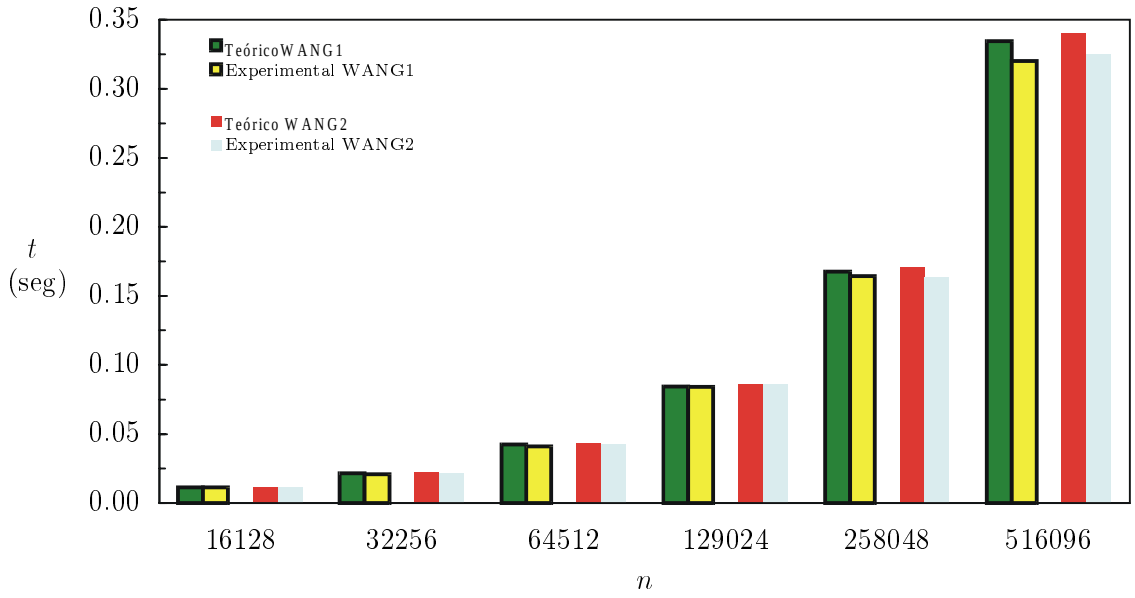
Figura 4.9: *Tiempos teóricos y experimentales de los algoritmos 4.2 (WANG1) y 4.3 (WANG2) en un IBM SP2 con 4 procesadores interconectados mediante switch.*

IBM SP2 6 procesadores *switch*				
	WANG1		WANG2	
n	Teórico	Experimental	Teórico	Experimental
126	0.0009	0.0013	0.0011	0.0016
252	0.0010	0.0011	0.0012	0.0014
504	0.0011	0.0011	0.0013	0.0013
1008	0.0015	0.0018	0.0017	0.0021
2016	0.0021	0.0019	0.0023	0.0020
4032	0.0034	0.0032	0.0036	0.0035
8064	0.0060	0.0058	0.0063	0.0060
16128	0.0112	0.0112	0.0116	0.0114
32256	0.0217	0.0210	0.0222	0.0213
64512	0.0425	0.0410	0.0434	0.0425
129024	0.0842	0.0842	0.0858	0.0858
258048	0.1677	0.1644	0.1707	0.1635
516096	0.3345	0.3203	0.3404	0.3252

Tabla 4.4: *Tiempos teóricos y experimentales de los algoritmos 4.2 (WANG1) y 4.3 (WANG2), medidos en un IBM SP2 con* 6 *procesadores interconectados mediante switch, para* $126 \leq n \leq 516096$.



(a) $126 \leq n \leq 8064$

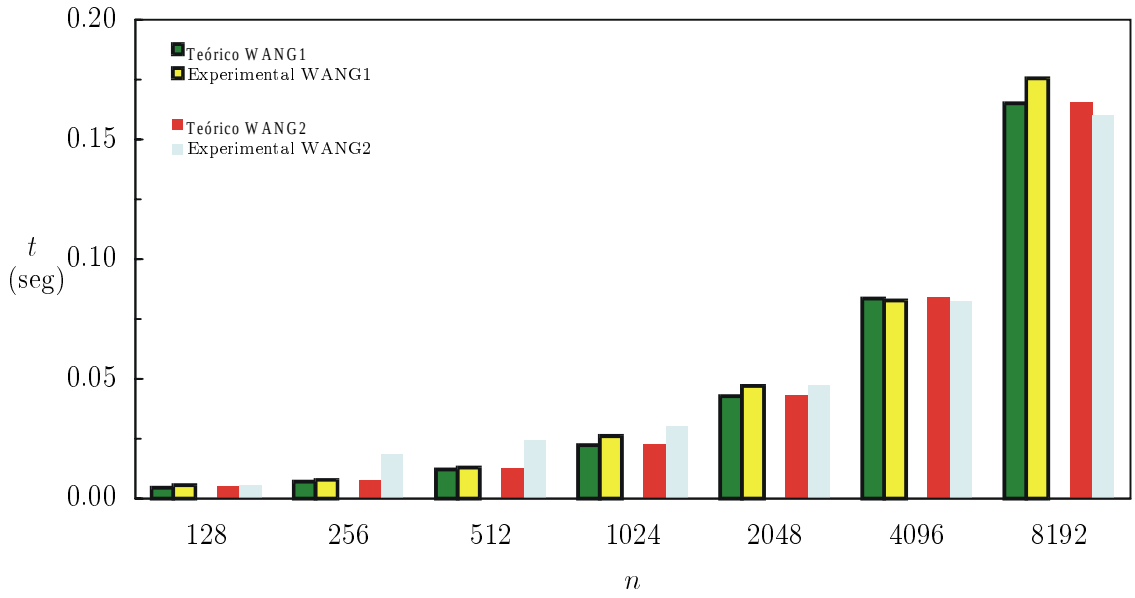


(b) $16128 \leq n \leq 516096$

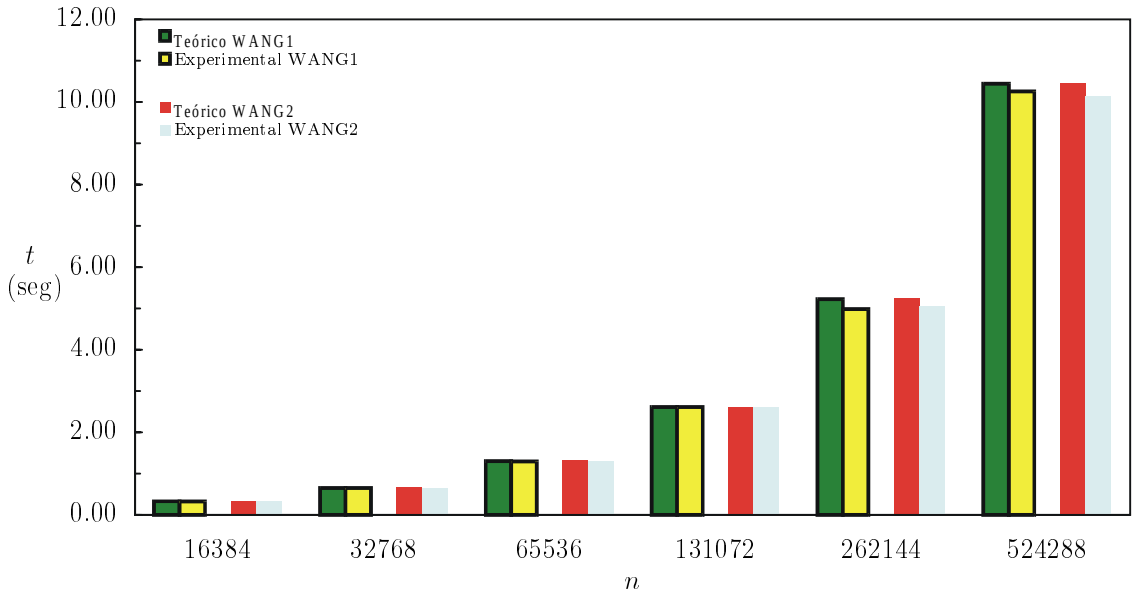
Figura 4.10: *Tiempos teóricos y experimentales de los algoritmos 4.2 (WANG1) y 4.3 (WANG2) en un IBM SP2 con 6 procesadores interconectados mediante switch.*

IBM SP2 2 procesadores *ethernet*				
	WANG1		WANG2	
n	Teórico	Experimental	Teórico	Experimental
128	0.0045	0.0056	0.0050	0.0054
256	0.0071	0.0078	0.0075	0.0186
512	0.0122	0.0130	0.0126	0.0245
1024	0.0224	0.0261	0.0228	0.0302
2048	0.0428	0.0470	0.0433	0.0474
4096	0.0835	0.0827	0.0841	0.0823
8192	0.1651	0.1756	0.1658	0.1601
16384	0.3282	0.3238	0.3292	0.3224
32768	0.6545	0.6506	0.6560	0.6520
65536	1.3070	1.2939	1.3096	1.2850
131072	2.6119	2.6161	2.6168	2.6161
262144	5.2219	4.9811	5.2311	5.0445
524288	10.4419	10.2511	10.4598	10.1243

Tabla 4.5: *Tiempos teóricos y experimentales de los algoritmos 4.2 (WANG1) y 4.3 (WANG2), medidos en un IBM SP2 con 2 procesadores interconectados mediante ethernet, para* $128 \leq n \leq 524288$.



(a) $128 \leq n \leq 8192$

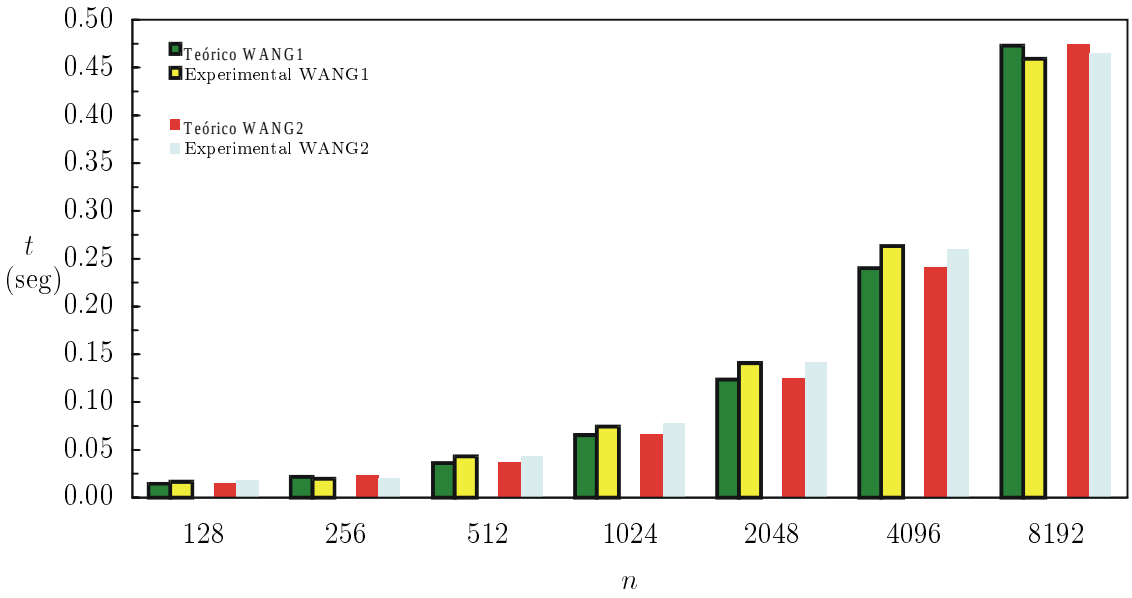


(b) $16384 \leq n \leq 524288$

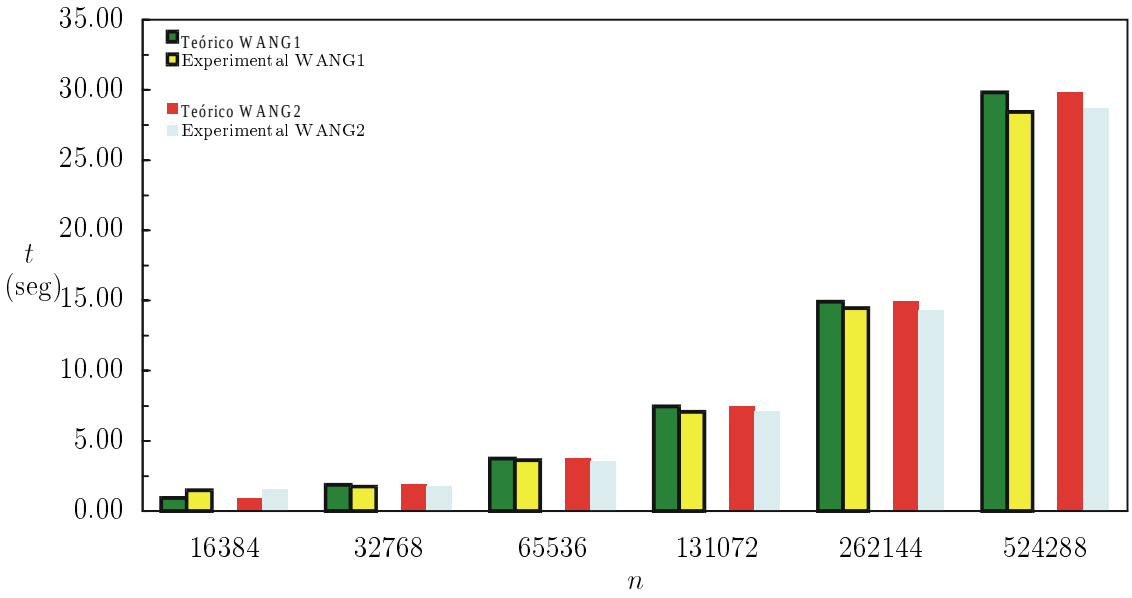
Figura 4.11: *Tiempos teóricos y experimentales de los algoritmos 4.2 (WANG1) y 4.3(WANG2) en un IBM SP2 con 2 procesadores interconectados mediante ethernet.*

IBM SP2 4 procesadores *ethernet*				
	WANG1		WANG2	
n	Teórico	Experimental	Teórico	Experimental
128	0.0145	0.0165	0.0156	0.0183
256	0.0217	0.0197	0.0229	0.0204
512	0.0363	0.0433	0.0374	0.0435
1024	0.0654	0.0744	0.0665	0.0782
2048	0.1236	0.1407	0.1248	0.1418
4096	0.2401	0.2634	0.2413	0.2598
8192	0.4730	0.4594	0.4743	0.4647
16384	0.9390	1.5034	0.9404	1.5297
32768	1.8708	1.7549	1.8725	1.7755
65536	3.7346	3.6297	3.7368	3.5466
131072	7.4620	7.0647	7.4653	7.1182
262144	14.9169	14.4628	14.9224	14.3265
524288	29.8266	28.4446	29.8365	28.7219

Tabla 4.6: *Tiempos teóricos y experimentales de los algoritmos 4.2 (WANG1) y 4.3 (WANG2), medidos en un IBM SP2 con* 4 *procesadores interconectados mediante ethernet, para* $128 \leq n \leq 524288$.



(a) $128 \leq n \leq 8192$

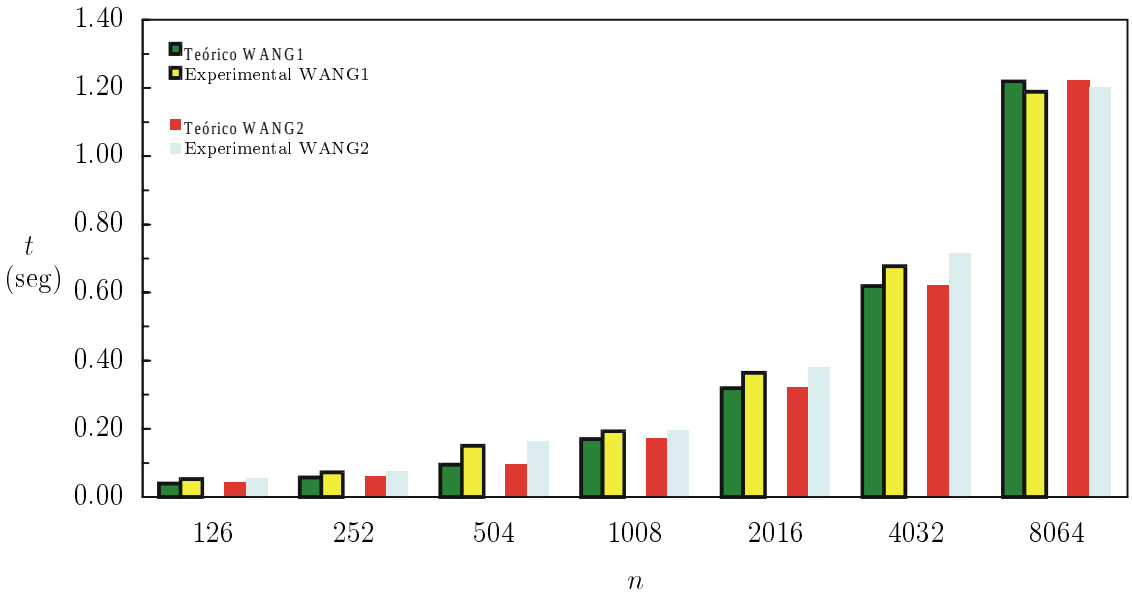


(b) $16384 \leq n \leq 524288$

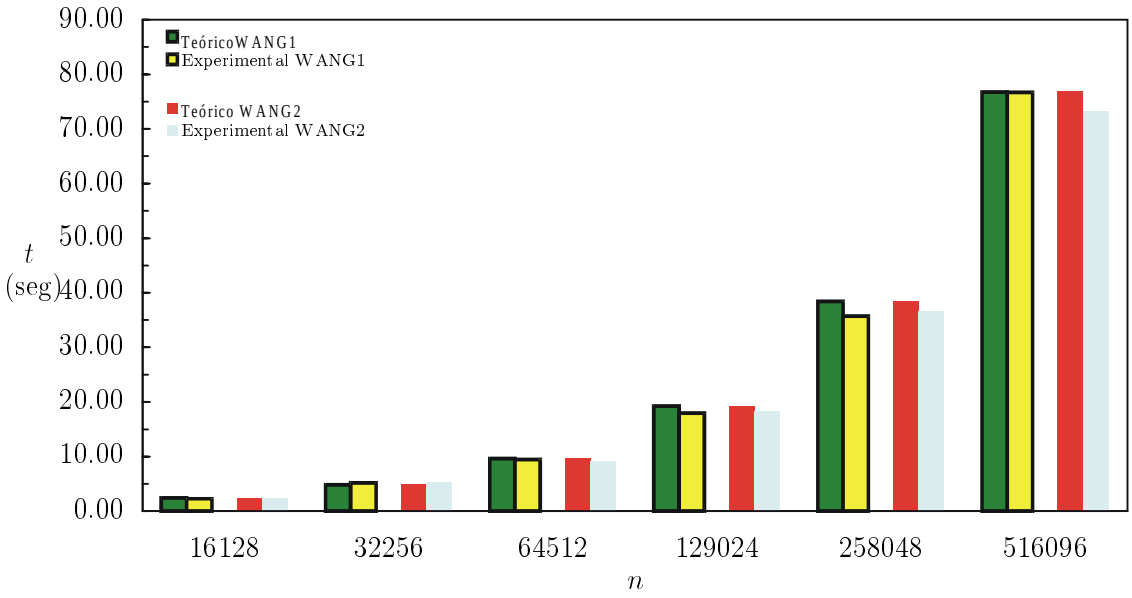
Figura 4.12: *Tiempos teóricos y experimentales de los algoritmos 4.2 (WANG1) y 4.3 (WANG2) en un IBM SP2 con 4 procesadores interconectados mediante ethernet.*

IBM SP2 6 procesadores *ethernet*				
	WANG1		WANG2	
n	Teórico	Experimental	Teórico	Experimental
126	0.0390	0.0523	0.0413	0.0541
252	0.0575	0.0729	0.0599	0.0752
504	0.0949	0.1504	0.0973	0.1631
1008	0.1698	0.1932	0.1722	0.1951
2016	0.3197	0.3647	0.3221	0.3812
4032	0.6195	0.6776	0.6220	0.7137
8064	1.2192	1.1891	1.2217	1.2026
16128	2.4186	2.2895	2.4212	2.3126
32256	4.8174	5.1402	4.8202	5.2438
64512	9.6149	9.4244	9.6181	9.1213
129024	19.2100	17.9577	19.2139	18.1939
258048	38.4002	35.7446	38.4055	36.5398
516096	76.7805	76.7245	76.7887	73.1317

Tabla 4.7: *Tiempos teóricos y experimentales de los algoritmos 4.2 (WANG1) y 4.3 (WANG2), medidos en un IBM SP2 con* 6 *procesadores interconectados mediante ethernet, para* $126 \leq n \leq 516096$.



(a) $126 \leq n \leq 8064$

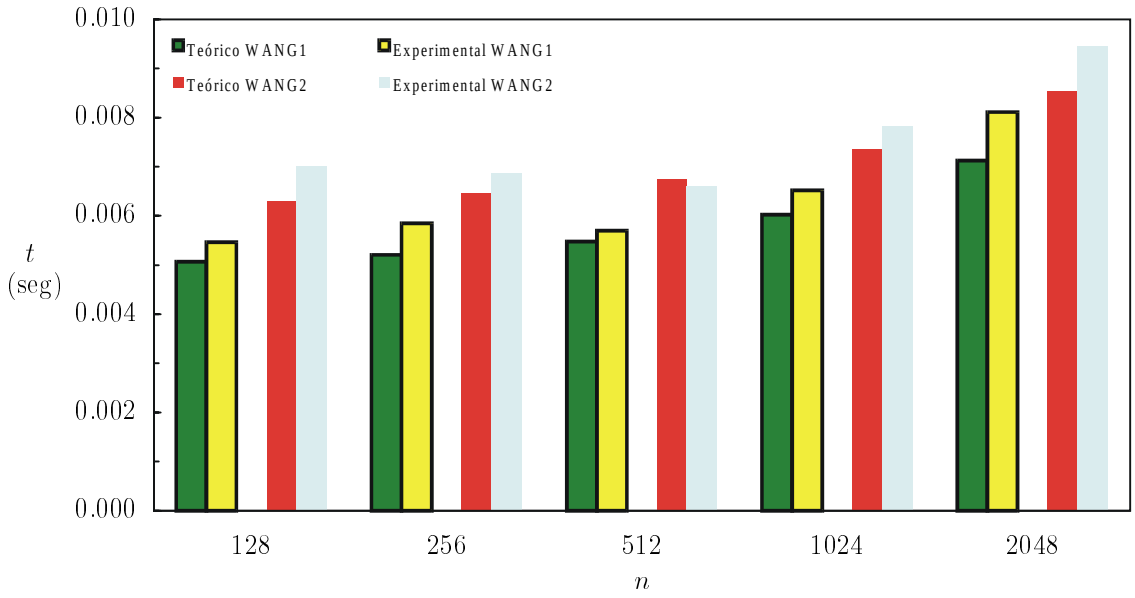


(b) $16128 \leq n \leq 516096$

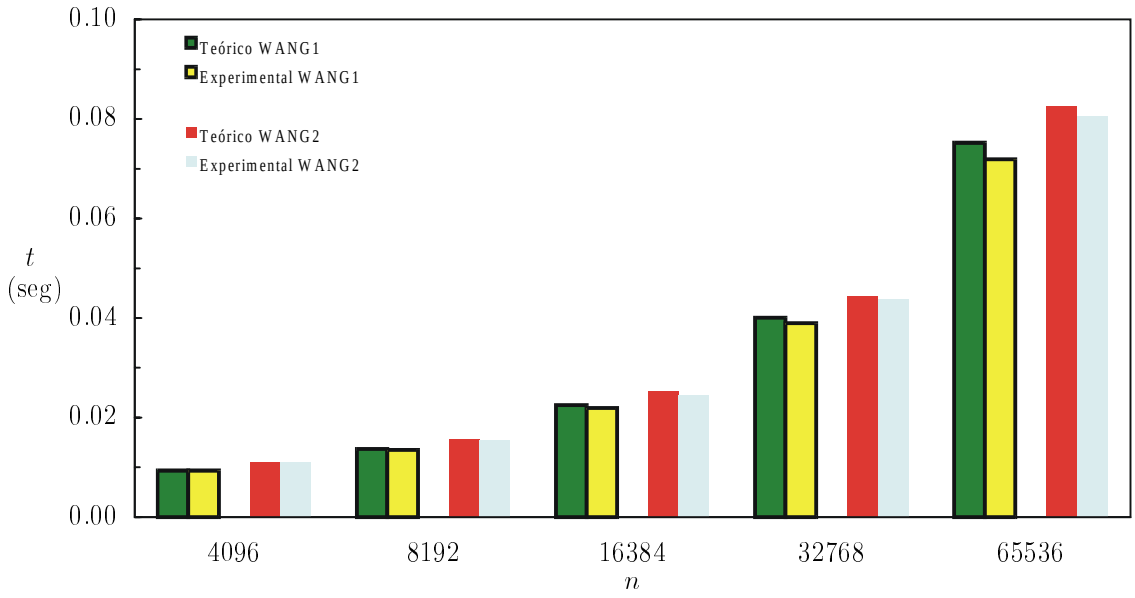
Figura 4.13: *Tiempos teóricos y experimentales de los algoritmos 4.2 (WANG1) y 4.3 (WANG2) en un IBM SP2 con 6 procesadores interconectados mediante ethernet.*

Cluster de PC's 2 procesadores				
	WANG1		WANG2	
n	Teórico	Experimental	Teórico	Experimental
128	0.0051	0.0055	0.0063	0.0070
256	0.0052	0.0058	0.0065	0.0069
512	0.0055	0.0057	0.0068	0.0066
1024	0.0060	0.0065	0.0074	0.0078
2048	0.0071	0.0081	0.0085	0.0095
4096	0.0093	0.0094	0.0109	0.0110
8192	0.0137	0.0135	0.0157	0.0154
16384	0.0225	0.0219	0.0252	0.0244
32768	0.0401	0.0390	0.0443	0.0438
65536	0.0752	0.0719	0.0824	0.0804

Tabla 4.8: *Tiempos teóricos y experimentales de los algoritmos 4.2 (WANG1) y 4.3 (WANG2) medidos en un cluster de PC's, para 2 procesadores y* $128 \leq n \leq 65536$.



(a) $128 \leq n \leq 2048$

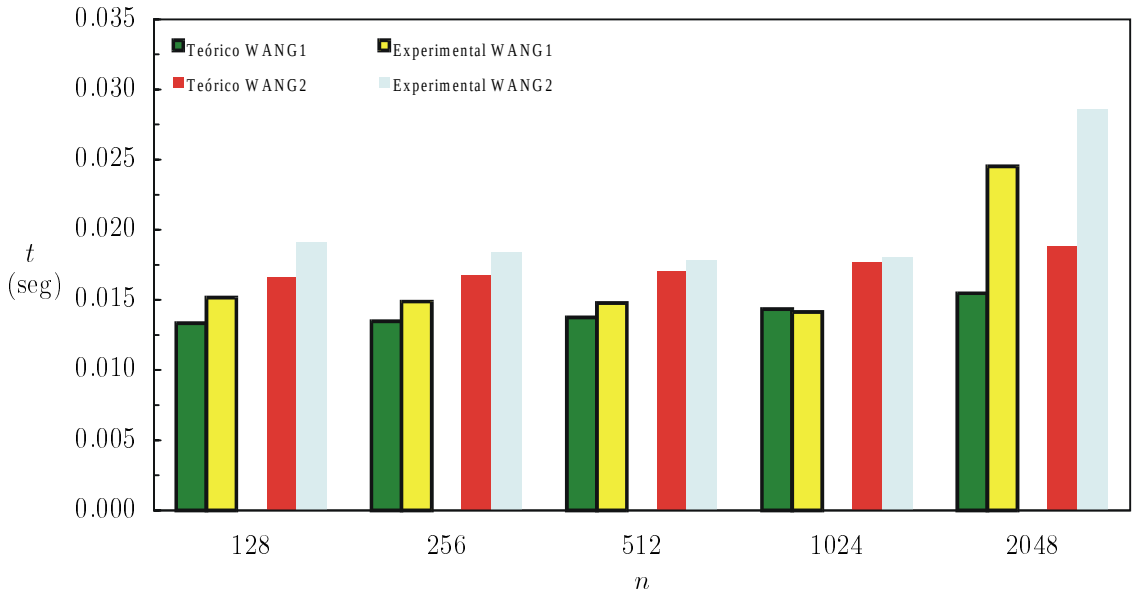


(b) $4096 \leq n \leq 65536$

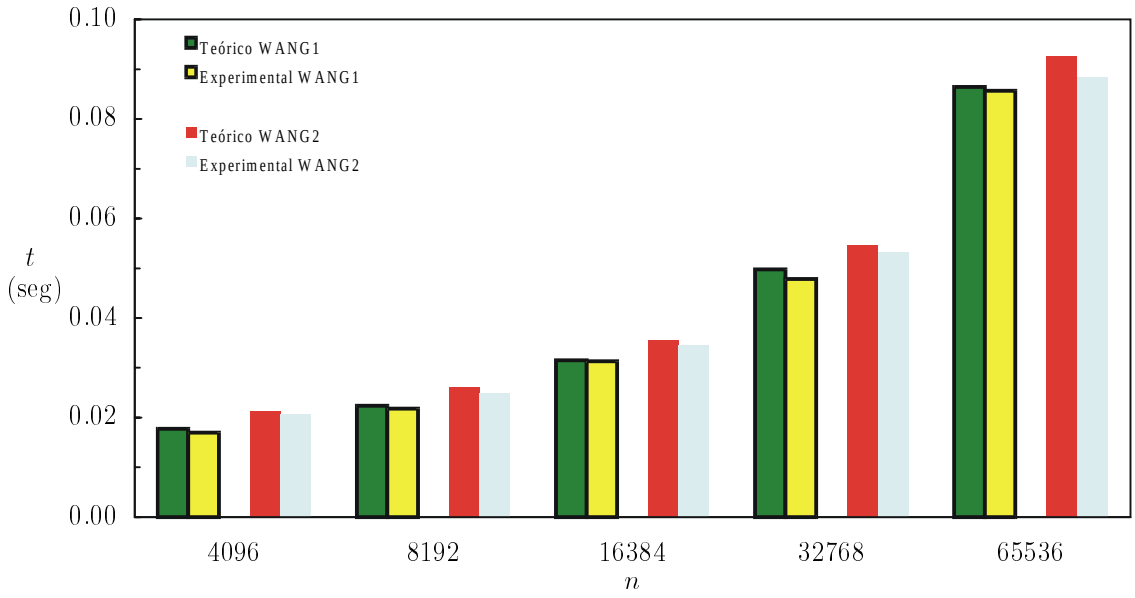
Figura 4.14: *Tiempos teóricos y experimentales de los algoritmos 4.2 (WANG1) y 4.3 (WANG2) en un cluster de PC's para 2 procesadores.*

Cluster de PC's 4 procesadores				
	WANG1		WANG2	
n	Teórico	Experimental	Teórico	Experimental
128	0.0133	0.0152	0.0166	0.0191
256	0.0135	0.0149	0.0168	0.0184
512	0.0138	0.0148	0.0171	0.0179
1024	0.0143	0.0141	0.0177	0.0180
2048	0.0155	0.0245	0.0189	0.0286
4096	0.0178	0.0170	0.0213	0.0207
8192	0.0224	0.0218	0.0260	0.0250
16384	0.0315	0.0313	0.0355	0.0346
32768	0.0498	0.0478	0.0546	0.0532
65536	0.0864	0.0857	0.0927	0.0884

Tabla 4.9: *Tiempos teóricos y experimentales de los algoritmos 4.2 (WANG1) y 4.3 (WANG2) medidos en un cluster de PC's, para 4 procesadores y* $128 \leq n \leq 65536$.



(a) $128 \leq n \leq 2048$

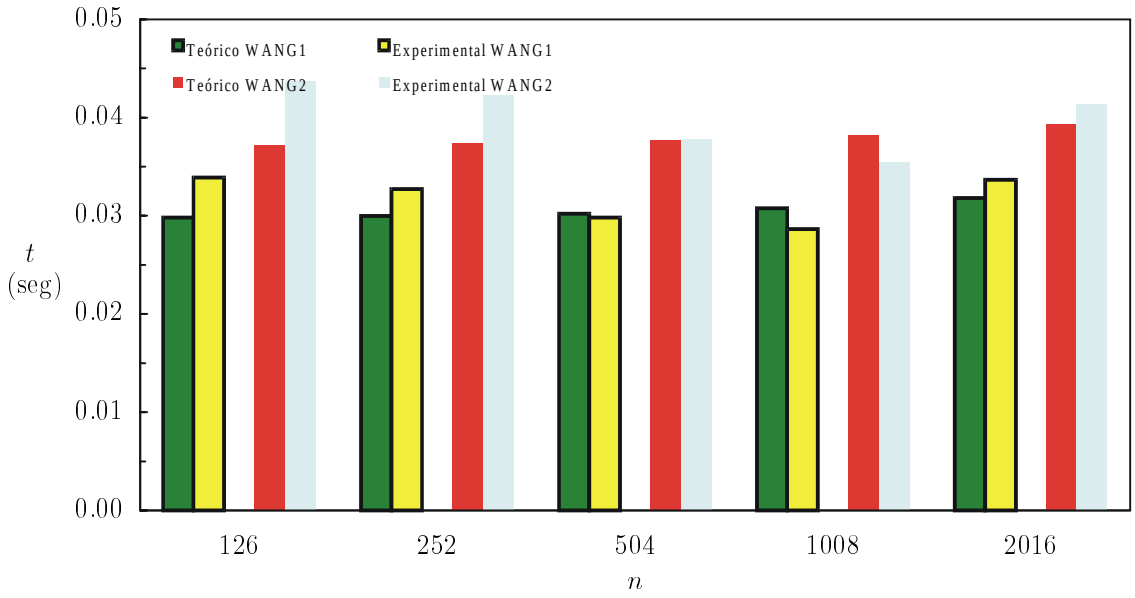


(b) $4096 \leq n \leq 65536$

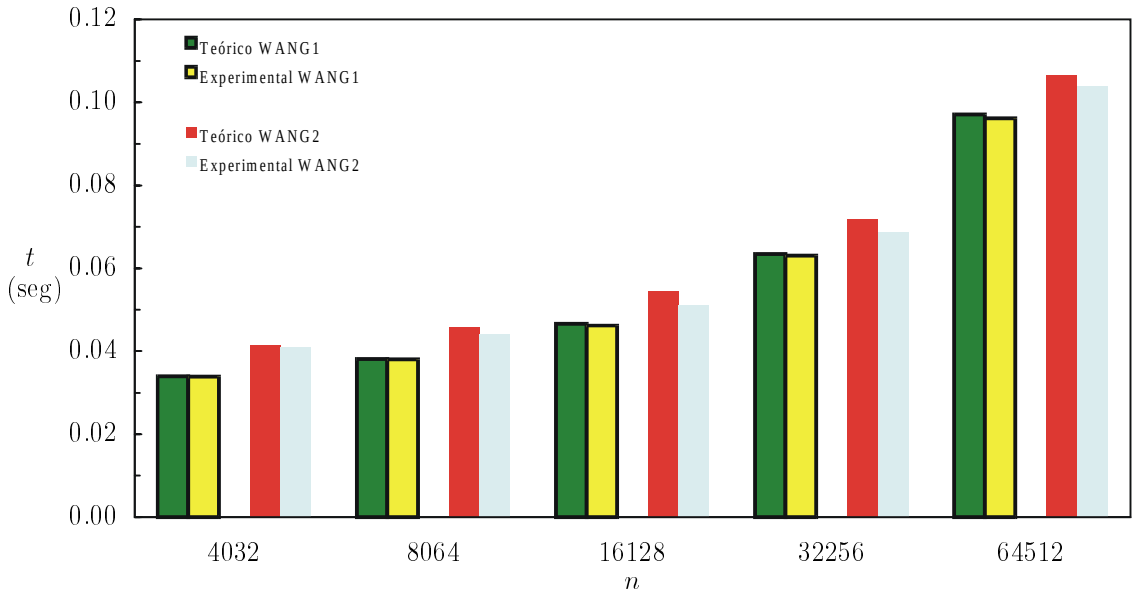
Figura 4.15: *Tiempos teóricos y experimentales de los algoritmos 4.2 (WANG1) y 4.3 (WANG2) en un cluster de PC's para 4 procesadores.*

Cluster de PC's 6 procesadores				
	WANG1		WANG2	
n	Teórico	Experimental	Teórico	Experimental
126	0.0298	0.0339	0.0373	0.0438
252	0.0300	0.0327	0.0374	0.0423
504	0.0302	0.0298	0.0377	0.0378
1008	0.0308	0.0286	0.0382	0.0355
2016	0.0318	0.0337	0.0393	0.0414
4032	0.0339	0.0339	0.0415	0.0408
8064	0.0381	0.0381	0.0458	0.0441
16128	0.0466	0.0462	0.0545	0.0511
32256	0.0634	0.0631	0.0718	0.0686
64512	0.0971	0.0962	0.1065	0.1038

Tabla 4.10: *Tiempos teóricos y experimentales de los algoritmos 4.2 (WANG1) y 4.3 (WANG2) medidos en un cluster de PC's, para* 6 *procesadores y* $126 \leq n \leq 64512$.

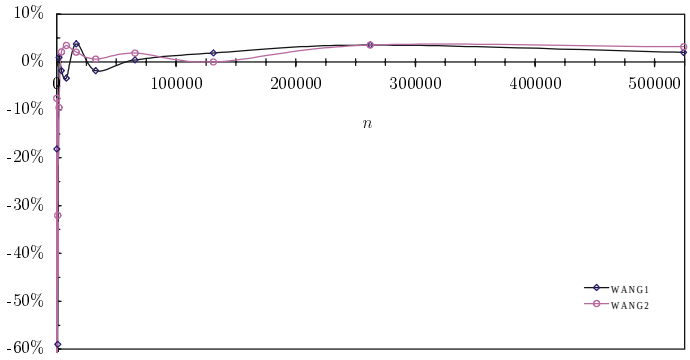


(a) $126 \leq n \leq 2016$

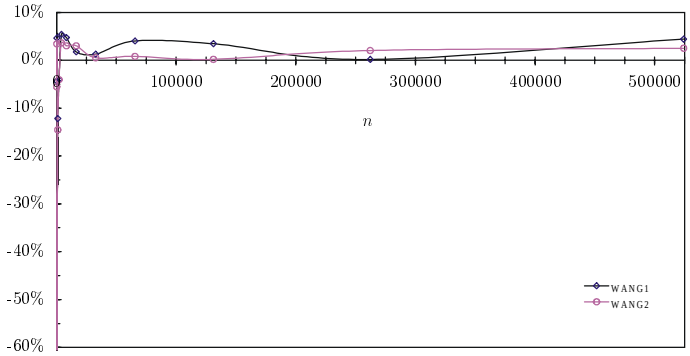


(b) $4032 \leq n \leq 64512$

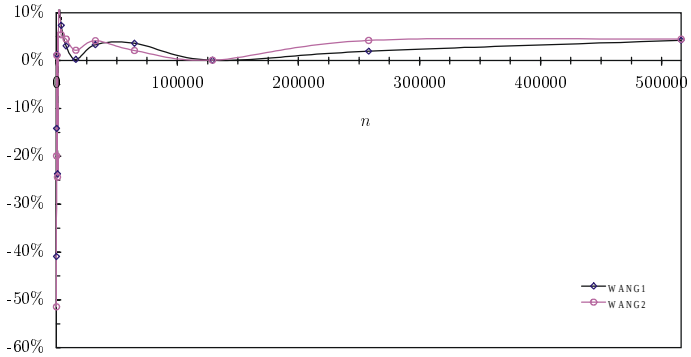
Figura 4.16: *Tiempos teóricos y experimentales de los algoritmos 4.2 (WANG1) y 4.3 (WANG2) en un cluster de PC's para* 6 *procesadores.*



(a) 2 *procesadores*

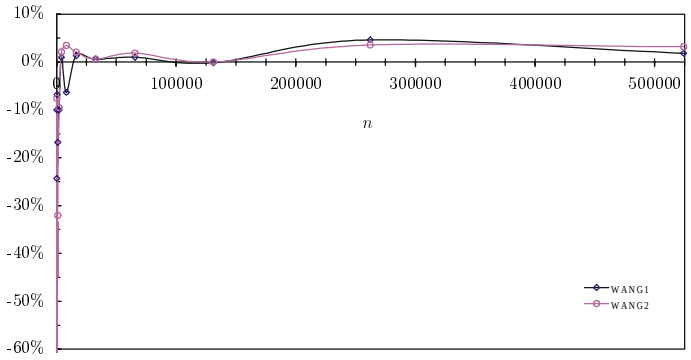


(b) 4 *procesadores*

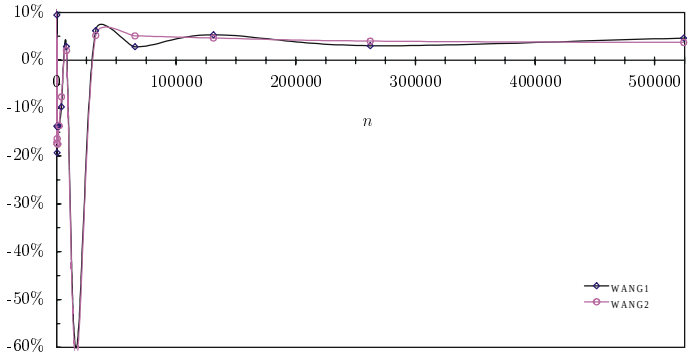


(c) 6 *procesadores*

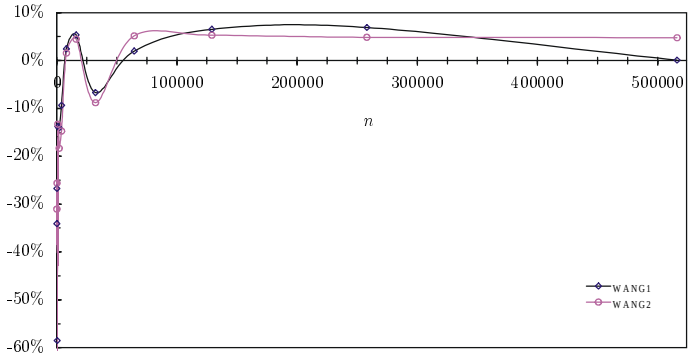
Figura 4.17: *Diferencias porcentuales entre los tiempos teóricos y experimentales de los algoritmos 4.2 (WANG1) y 4.3 (WANG2) en un IBM SP2 con switch.*



(a) 2 *procesadores*

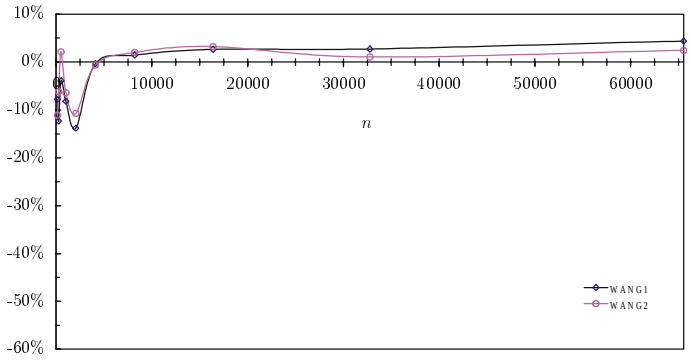


(b) 4 *procesadores*

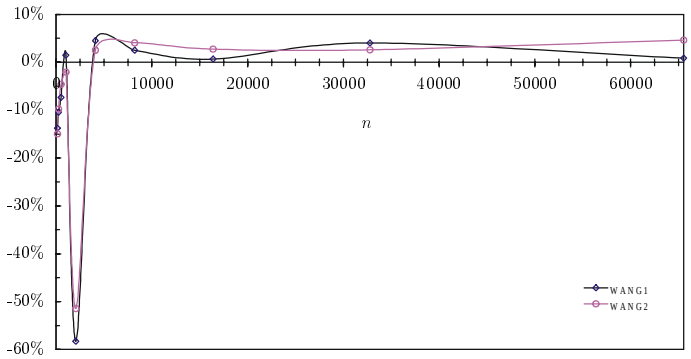


(c) 6 *procesadores*

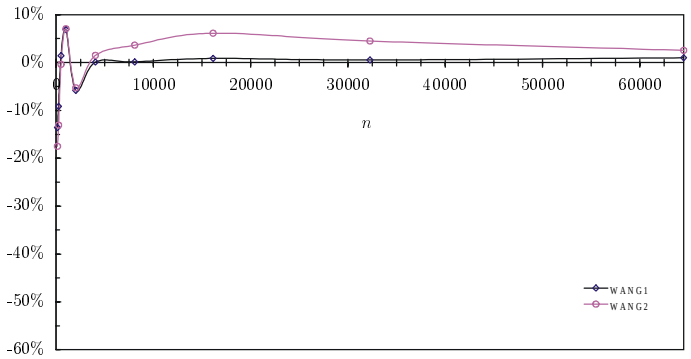
Figura 4.18: *Diferencias porcentuales entre los tiempos teóricos y experimentales de los algoritmos 4.2 (WANG1) y 4.3 (WANG2) en un IBM SP2 con ethernet.*



(a) 2 *procesadores*



(b) 4 *procesadores*



(c) 6 *procesadores*

Figura 4.19: *Diferencias porcentuales entre los tiempos teóricos y experimentales de los algoritmos 4.2 (WANG1) y 4.3 (WANG2) en un cluster de PC's.*

CRAY T3D 256 procesadores			
n	WANG1	WANG2	Mejor
512	0.0019	0.0016	WANG2
1024	0.0018	0.0016	WANG2
2048	0.0020	0.0018	WANG2
4096	0.0024	0.0023	WANG2
8192	0.0034	0.0033	WANG2
16384	0.0055	0.0054	WANG2
32768	0.0097	0.0096	WANG2
65536	0.0180	0.0180	WANG2
131072	0.0347	0.0348	WANG1
262144	0.0681	0.0683	WANG1
524288	0.1350	0.1354	WANG1

Tabla 4.11: *Tiempos teóricos de los algoritmos 4.2 (WANG1) y 4.3 (WANG2) medidos en un CRAY T3D, para* 256 *procesadores y* $512 \leq n \leq 524288$.

IBM SP2 *switch*				
	WANG1		WANG2	
p	S_p	E_p	S_p	E_p
2	0.38	19.19%	0.36	17.90%
4	0.29	7.18%	0.28	6.99%
6	0.27	4.57%	0.27	4.49%

(a) *IBM SP2 switch*

IBM SP2 *ethernet*				
	WANG1		WANG2	
p	S_p	E_p	S_p	E_p
2	0.01	0.45%	0.01	0.45%
4	0.00	0.08%	0.00	0.08%
6	0.00	0.02%	0.00	0.02%

(b) *IBM SP2 ethernet*

Cluster de PC's				
	WANG1		WANG2	
p	S_p	E_p	S_p	E_p
2	0.42	21.20%	0.39	19.35%
4	0.37	9.22%	0.34	8.60%
6	0.32	5.39%	0.29	4.91%

(c) *Cluster de PC's*

Tabla 4.12: *Speed-up (S_p) y eficiencia (E_p) de los algoritmos 4.2 (WANG1) y 4.3 (WANG2) en un IBM SP2 y un cluster de PC's.*

s	p	l	g	$n_{\frac{1}{2}}$
12	1	68	0.3	94
	2	164	0.7	71
	4	168	0.7	66
	8	175	0.8	59
	16	181	0.9	61
	32	201	1.1	28
	64	148	1	27
	128	301	1.1	20
	256	387	1.2	15

(a) *CRAY T3D*

s	p	l	g	$n_{\frac{1}{2}}$
46.7	1	86	2.12	9
	2	269	0.87	33
	4	357	0.87	40
	8	506	0.81	40
	16	751	1.04	38
	32	1252	1.31	45

(b) *CRAY T3E*

Tabla 4.13: *Valores de parámetros BSP.*

CRAY T3D				
	WANG1		WANG2	
p	S_p	E_p	S_p	E_p
2	0.88	44.01%	0.76	37.77%
4	1.38	34.40%	1.22	30.47%
8	2.00	24.98%	1.83	22.84%
16	2.47	15.43%	2.33	14.59%
32	2.47	7.73%	2.40	7.51%
64	2.91	4.54%	2.86	4.47%
128	2.77	2.16%	2.75	2.15%
256	2.59	1.01%	2.58	1.01%

(a) *CRAY T3D*

CRAY T3E				
	WANG1		WANG2	
p	S_p	E_p	S_p	E_p
2	0.79	39.64%	0.69	34.51%
4	1.30	32.60%	1.16	29.05%
8	1.99	24.83%	1.82	22.71%
16	2.24	14.00%	2.13	13.30%
32	1.90	11.88%	1.82	11.37%

(b) *CRAY T3E*

Tabla 4.14: *Speed-up (S_p) y eficiencia (E_p) de los algoritmos 4.2 (WANG1) y 4.3 (WANG2) en un CRAY T3D y un CRAY T3E.*