UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DESCOMPOSICIÓN DE DOMINIO CON PROGRAMACIÓN EN PARALELO APLICADA A LA SIMULACIÓN NUMÉRICA DE YACIMIENTOS

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO PETROLERO

P R E S E N T A N :

CEDILLO TREJO URIEL
ORANTES LÓPEZ RODRIGO

DIRECTOR DE TESIS:
Dr. Víctor Hugo Arana Ortiz.

CODIRECTOR DE TESIS:
Mat. Luis Alberto Vázquez Maison.

MÉXICO, D. F. Diciembre 2005.

Agradecimientos

Al Grupo de Simulación Numérica de Yacimientos de la Facultad de Ingeniería de la UNAM, por las facilidades otorgadas para realización de este trabajo.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

La Simulación Numérica de Yacimientos es una herramienta muy utilizada e importante en la Industria Petrolera. A medida que el desarrollo de algoritmos, software y hardware se desarrollan, también existe la necesidad de simular problemas más grandes y en mayor detalle para realizar pronósticos más reales y mejorar el desarrollo de los yacimientos. Lo anterior aumenta los requerimientos de cómputo. Entre las técnicas para reducir los tiempos de cómputo a una escala práctica están la descomposición de dominio (DD) y la programación en paralelo (PP).

La Descomposición de Dominio debe de ser paralelizable, por lo tanto al considerar la DD se esta hablando automáticamente de programación en paralelo. La técnica de DD permite particionar problemas grandes y difíciles en problemas más pequeños y simples de resolver, logrando con esto resolver problemas muy complejos o grandes en forma correcta y rápida.

El objetivo de este trabajo es doble: el primero consiste en aplicar la Descomposición de Dominio utilizando un simulador semi-implícito de aceite negro, monofásico y en 3D para ver qué tan factible es el uso de esta técnica en la Simulación Numérica de Yacimientos. Y el segundo es analizar los beneficios que se obtienen al optimizarla con la Programación en Paralelo.

De esta forma, se da a conocer: en el primer capítulo, los fundamentos de la simulación numérica de yacimientos; en el segundo y tercer capítulos, los conceptos básicos de la Descomposición de Dominio y de la Programación en Paralelo, respectivamente; en el cuarto capítulo, el trabajo realizado al aplicar la Descomposición de Dominio en el simulador, así como los resultados obtenidos al paralelizar el simulador; en el quinto capítulo se muestran algunas aplicaciones que se hicieron con el simulador y; en el sexto capítulo se dan las conclusiones obtenidas de este trabajo y las recomendaciones para mejorar su desempeño.

INTRODUCCIÓN.

En general, el término "simulación" se refiere a la representación de algunos procesos o fenómenos a través de modelos teóricos y/o físicos. En la Industria Petrolera, en particular en la Ingeniería de Yacimientos, al área que realiza esto se le conoce como: Área de Simulación Numérica de Yacimientos.

La Simulación es una herramienta muy usada en todas las áreas de la Ingeniería y de la Ciencia, ya que, si es bien aprovechada, permite mejorar el rendimiento, tanto técnico como económico, de los procesos que se analicen. Esto se logra al visualizar el comportamiento de los fenómenos involucrados ante diversos estímulos (causas) o esquemas de operación con el fin de encontrar aquel que nos proporcione los mejores resultados (los mejores efectos). La ventaja de todo esto, es que los errores que se cometen no tienen repercusión en la realidad, pues, fueron hechos en un simulador y no en el proceso o fenómeno real. Sin embargo, la experiencia adquirida con las simulaciones sí tiene impacto en la realidad, pues, brinda valiosa información del proceso o fenómeno estudiado.

Sin embargo, algo que no hay que perder de vista en la Simulación es que todo lo que obtendrá será resultado de los datos que se ingresen al simulador. Por eso, el juicio de quien usa el simulador es muy importante, pues, debe decidir qué datos se van a usar y los límites de calidad permisibles de los mismos. Y, en caso de que no se tengan los suficientes, habrá que decidir cuáles deberán ser tomados y de qué forma, para obtener buenos resultados y no afectar la rentabilidad. Además, se debe ser capaz de simular los diferentes escenarios que se quieren e interpretar los datos arrojados por éstos.

En la Simulación Numérica de Yacimientos, cada modelo debe ser adaptado al yacimiento que se quiere estudiar, pues, un simulador sólo es útil cuando se ajusta al caso de campo. Para esto existen modelos en coordenadas cartesianas, cilíndricas y esféricas, así como, en una, dos o tres dimensiones. También, existen modelos monofásicos -para aceite negro o gas- y multifásicos. En resumen, depende de quien realiza el estudio escoger el modelo que mejor se adapte a su problema.

El principal propósito de la Simulación Numérica de Yacimientos es predecir el ritmo de recuperación de los hidrocarburos para diferentes escenarios de operación de campo. Por eso, es muy importante que los resultados del simulador se acercen a la realidad.

Algo a lo que hay que dedicar atención especial es la selección del número de celdas de la malla de simulación y el tamaño de las mismas, pues, esto influye fuertemente en el tiempo de corrida y en el detalle de los resultados obtenidos. Mientras más celdas se usen y de menor tamaño mejor será el detalle de la descripción del fenómeno, pero, así también crecerá el tiempo de simulación.

A pesar del tiempo invertido y los cuidados que se deben tener en la simulación para alcanzar resultados satisfactorios, los beneficios obtenidos, tanto en tiempo como en dinero, valen la pena. Debido a esto, ha habido bastante interés en el avance de la simulación, sobre todo, después de mediados de los 40´s, cuando las computadoras electrónicas fueron una realidad. Desde entonces, la simulación con modelos numéricos ha ido ganando terreno debido a que, aunque no era nueva (los principios y ecuaciones de la Simulación Numérica de Yacimientos ya existían), sí alcanzó su mejor rendimiento con este adelanto tecnológico.

El incremento en el uso de la simulación numérica se ha visto beneficiado con los constantes avances en la capacidad y velocidad de cómputo de las máquinas, así como, en el desarrollo de algoritmos numéricos más eficientes. Sin embargo, esto es también una limitante, pues, el alcance de la simulación está ligado con estos adelantos. Por eso, se han y se siguen realizando fuertes inversiones para adelantar este límite, desarrollando mejores algoritmos de solución y máquinas más potentes.

Uno de los principales esfuerzos, que se están haciendo, en el área de optimización de algoritmos, para aumentar el alcance de la simulación y su capacidad, es el de la Descomposición de Dominio (DD). Ésta es una técnica que permite descomponer un dominio en subdominios independientes, más pequeños y, generalmente, menos complejos, los cuales, una vez analizados, son vueltos a juntar para obtener la solución total.

Algunas de las ventajas que ofrece la Descomposición de Dominio son: la capacidad de cómputo de las máquinas aumenta, pues, se analiza cada subdominio de manera independiente y no el dominio total; se pueden resolver problemas más grandes y con más detalle; es paralelizable, o sea, puede usarse con Programación en Paralelo y; al usarse con Programación en Paralelo el tiempo de simulación se optimiza.

Otro esfuerzo que se está realizando, pero, en este caso, para aumentar la potencia de las máquinas, es la Programación en Paralelo. Ésta consiste en poner a trabajar varios procesadores simultáneamente, para que se repartan el trabajo y terminen en menos tiempo. Cada procesador toma la información que le corresponde, trabaja con ella y, luego, la comparte con los demás, y así sucesivamente, hasta que llegan al resultado final.

La Programación en Paralelo ofrece algunas ventajas como: permite reducir la carga de trabajo de las máquinas al distribuirla en varias en vez de una sola; permite optimizar el tiempo de ejecución de programas robustos; se puede reducir la complejidad de algunos programas y; se optimizan los recursos de cómputo, alcanzándose, e incluso superándose, el rendimiento de las supercomputadoras con máquinas paralelas más económicas.

El objetivo de esta tesis es aplicar la Descomposición de Dominio en un simulador semi-implícito de aceite negro, monofásico y en 3D para ver qué tan factible es el uso de esta técnica en la Simulación Numérica de Yacimientos y si los resultados que se obtienen son satisfactorios.

La estructura de este trabajo es la siguiente: en el primer capítulo se muestran los fundamentos de la Simulación Numérica de Yacimientos; en el segundo y tercer capítulos, los conceptos básicos de la Descomposición de Dominio y de la Programación en Paralelo, respectivamente; en el cuarto capítulo, el trabajo realizado y los resultados obtenidos al aplicar la Descomposición de Dominio y la Programación en Paralelo en el simulador; en el quinto capítulo se muestran algunas aplicaciones que se hicieron con el simulador y; en el sexto capítulo, y último, se dan las conclusiones obtenidas de este trabajo y algunas recomendaciones para mejorar su aplicación.

CAPÍTULO I. FUNDAMENTOS DE LA SIMULACIÓN NUMÉRICA DE YACIMIENTOS.

Antes de describir la técnica de Descomposición de Dominio es importante conocer algunos aspectos de la Simulación Numérica de Yacimientos, SNY. A continuación se discute brevemente las bases teóricas de la SNY.

1.1) Flujo Monofásico de un Fluido Ligeramente Compresible.

En la ingeniería de yacimientos se utiliza la ecuación de difusión para describir el flujo de los fluidos en los medios porosos. Ésta se obtiene al combinar la Ley de conservación de masa (ecuación de continuidad), la Ley de conservación de momento (Ley de Darcy), la ecuación de estado y la ley de conservación de energía (Aziz y Setari, 1979). Sin embargo, como el flujo de fluidos en el yacimiento se considera a temperatura constante, la ecuación de conservación de energía, frecuentemente, no es considerada. En el caso de un flujo de una fase en una dimensión y con un fluido de baja compresibilidad, la ecuación de difusión es:

$$\frac{\partial^2 p}{\partial x^2} = \frac{\mu}{k}\left[\phi c_t \frac{\partial p}{\partial t} + \tilde{q}_s\right] \qquad \textbf{…(1.1)}$$

y, para tres dimensiones:

$$\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} + \frac{\partial^2 p}{\partial z^2} = \frac{\mu}{k}\left[\phi c_t \frac{\partial p}{\partial t} + \tilde{q}_s\right] \qquad \textbf{…(1.2.a)}$$

O utilizando el operador laplaciano,

$$\nabla^2 p = \frac{\mu}{k}\left[\phi c_t \frac{\partial p}{\partial t} + \tilde{q}_s\right] \qquad \textbf{…(1.2.b)}$$

La **ecuación 1.2.b** es general para cualquier sistema de coordenadas, (Ertekin y otros 2001**)** y es la base de la teoría de pruebas de presión.

1.2) Discretización de la Ecuación de Difusión.

En casos donde la permeabilidad y la viscosidad son función del espacio, la **ecuación 1.2** toma la siguiente forma:

$$\frac{\partial}{\partial x}\left(\frac{k_x}{\mu}\frac{\partial p}{\partial x}\right)+\frac{\partial}{\partial y}\left(\frac{k_y}{\mu}\frac{\partial p}{\partial y}\right)+\frac{\partial}{\partial z}\left[\frac{k_z}{\mu}\left(\frac{\partial p}{\partial z}-\rho g\right)\right]-\tilde{q}_s=\phi c_t\frac{\partial p}{\partial t} \qquad \textbf{…(1.2.c)}$$

Debido a que la **ecuación 1.2.c** es altamente no lineal, existen soluciones analíticas para casos muy simples, cuando la **ecuación 1.2.c** no es aplicable en problemas más reales, se recurre a algoritmos numéricos para su solución. Al usar algoritmos numéricos, se trabaja en un dominio discreto en vez de un dominio continuo. Por eso, cuando se transforma una ecuación analítica a su forma numérica, para ser resuelta, se dice que se ha discretizado.

En este trabajo, no se muestra el desarrollo de la ecuación de difusión ni su proceso de discretización por diferencias finitas, así como, tampoco se profundiza en el estudio de los diferentes tipos de mallas de simulación y su generación. Debido a que no es el objetivo de la tesis. Sin embargo, en las referencias se mencionan las fuentes que se consultaron y que se consideran importantes **(Hernández R, O. Y Del Valle M, P., 2005)**.

Al pasar del dominio continuo al dominio discreto se genera una malla conocida como malla de simulación, los puntos centrales de cada celda son los puntos en donde se desea conocer la variación de la incógnita con el tiempo y espacio. Esta malla representa al dominio físico que se va a simular y está compuesta por un número finita de celdas. La cantidad de estas celdas y el número de incógnitas en el sistema determina el tamaño del sistema de ecuaciones que se generará. Mientras mayor sea el número de nodos, mayor será el tamaño del sistema generado, así como, el tiempo de solución requerido.

Las mallas pueden estar en coordenadas cartesianas, cilíndricas o esféricas, todo depende de la geometría del dominio que se desea analizar. También, pueden generarse en una, dos o tres dimensiones, dependiendo del número de direcciones en las que se desee conocer la variación de las incógnitas. El simulador empleado utiliza mallas en coordenadas cartesianas y en tres dimensiones **(Hernández y Del Valle, 2005).**

La **figura 1.1** es un ejemplo de una malla de simulación, de 24 nodos, en coordenadas cartesianas y en tres dimensiones, compuesta por 4 nodos en *i*, 3 en *j* y 2 en *k*.

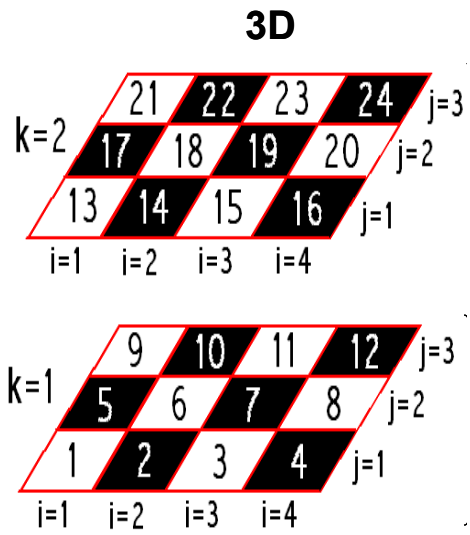


Figura 1.1. Ejemplo de malla de simulación en coordenadas cartesianas en 3D. Cada plano *XY* tiene un espesor Δz.

Existen varios métodos de discretización, uno de los más usados en la Simulación Numérica de Yacimientos es el Método de Diferencias Finitas (MDF). Este método utiliza la serie de Taylor truncada para aproximar las derivadas de la **ecuación 1.2.c**, en tiempo y en espacio.

En el simulador que se esta usando, se utilizaron diferencias centrales para discretizar en el espacio y diferencias progresivas para discretizar en el tiempo. La formulación es implícita en presión, es decir, son calculadas a un tiempo *n+1*. La transmisibilidad se evaluó al

tiempo *n*. De esta forma, una vez discretizada la **ecuación 1.2.c** y desarrollando y reagrupando términos se obtiene **(Hernández R, O. Y Del Valle M, P., 2005)**:

$$l_{i,j,k}p_{i,j,k-1}^{n+1}+s_{i,j,k}p_{i,j-1,k}^{n+1}+o_{i,j,k}p_{i-1,j,k}^{n+1}+c_{i,j,k}p_{i,j,k}^{n+1}+e_{i,j,k}p_{i+1,j,k}^{n+1}$$
$$+n_{i,j,k}p_{i,j+1,k}^{n+1}+u_{i,j,k}p_{i,j,k+1}^{n+1}=d_{i,j,k}-q_{i,j,k}^{n} \qquad \textbf{…(1.3)}$$

Esta nueva ecuación numérica, está compuesta por el término fuente *q*, el coeficiente *d* y siete coeficientes (*l, s, o, c, e, n, u, d*), los cuales resultan de la interacción que tiene el nodo, para el que se realizan los cálculos, con sus seis caras, como se muestra en la **figura 1.2**.

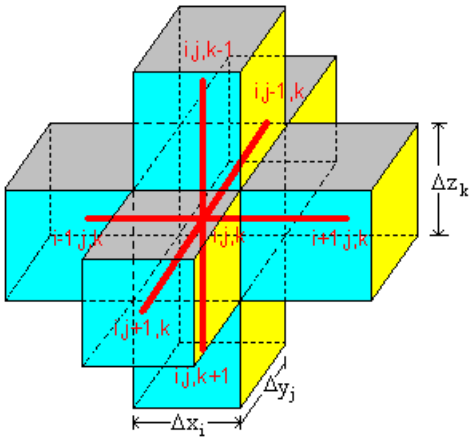


Figura 1.2. Nodo de análisis i, j, k y sus nodos vecinos.

El término fuente *q* tiene un gasto de inyección (+) o producción (-), en caso de que hubiera algún pozo en el nodo que se está analizando, el coeficiente *d* y los siete coeficientes dependen de las transmisibilidades al tiempo *n*, que existen entre el nodo de interés y sus seis fronteras, por lo tanto, son conocidos. Estos coeficientes están definidos como:

$$l_{i,j,k} = T^n_{i,j,k-\frac{1}{2}}$$

$$s_{i,j,k} = T^n_{i,j-\frac{1}{2},k}$$

$$o_{i,j,k} = T^n_{i-\frac{1}{2},j,k}$$

$$c_{i,j,k} = -\left[T^n_{i,j,k-\frac{1}{2}} + T^n_{i,j-\frac{1}{2}} + T^n_{i-\frac{1}{2},j} + T^n_{i+\frac{1}{2},j} + T^n_{i,j+\frac{1}{2}} + T^n_{i,j,k+\frac{1}{2}} + \frac{Vr_{i,j,k}}{\Delta t}\left\{ \phi^n_{i,j,k}\left(b_o c_f \right)_{i,j,k} + b^n_{i,j,k}\left(\phi_o c_r \right)_{i,j,k} \right\} \right]$$

$$e_{i,j,k} = T^n_{i+\frac{1}{2},j,k}$$

$$n_{i,j,k} = T^n_{i,j+\frac{1}{2},k}$$

$$u_{i,j,k} = T^n_{i,j,k+\frac{1}{2}}$$

$$d_{i,j,k} = -\frac{Vr_{i,j,k}}{\Delta t}\left[\phi^n_{i,j,k}\left(b_o c_f \right)_{i,j,k} + b^n_{i,j,k}\left(\phi_o c_r \right)_{i,j,k} \right] p^n_{i,j,k} + T^n_{i,j,k+\frac{1}{2}}\gamma_{i,j,k+\frac{1}{2}}\left(D_{i,j,k+1} - D_{i,j,k} \right) - T^n_{i,j,k-\frac{1}{2}}\gamma_{i,j,k-\frac{1}{2}}\left(D_{i,j,k} - D_{i,j,k-1} \right)$$

donde:

$$T^n_{i+\frac{1}{2},j,k} = \left(\frac{\Delta y \Delta z}{\Delta x} \right)_{i+\frac{1}{2},j,k} \left(\frac{k}{\mu B} \right)_{i+\frac{1}{2},j,k} \qquad T^n_{i-\frac{1}{2},j,k} = \left(\frac{\Delta y \Delta z}{\Delta x} \right)_{i-\frac{1}{2},j,k} \left(\frac{k}{\mu B} \right)_{i-\frac{1}{2},j,k}$$

$$T^n_{i,j+\frac{1}{2},k} = \left(\frac{\Delta x \Delta z}{\Delta y} \right)_{i,j+\frac{1}{2},k} \left(\frac{k}{\mu B} \right)_{i,j+\frac{1}{2},k} \qquad T^n_{i,j-\frac{1}{2},k} = \left(\frac{\Delta x \Delta z}{\Delta y} \right)_{i,j-\frac{1}{2},k} \left(\frac{k}{\mu B} \right)_{i,j-\frac{1}{2},k}$$

$$T^n_{i,j,k+\frac{1}{2}} = \left(\frac{\Delta x \Delta y}{\Delta z} \right)_{i,j,k+\frac{1}{2}} \left(\frac{k}{\mu B} \right)_{i,j,k+\frac{1}{2}} \qquad T^n_{i,j,k-\frac{1}{2}} = \left(\frac{\Delta x \Delta y}{\Delta z} \right)_{i,j,k-\frac{1}{2}} \left(\frac{k}{\mu B} \right)_{i,j,k-\frac{1}{2}}$$

La **ecuación 1.3** representa un sistema del tipo ***Ax = b***, en donde; ***A*** es una matriz heptadiagonal, formada por los siete coeficientes (*l, s, o, c, e, n, u*), y de dimensiones n x n (en donde n es el número de nodos que conforman la malla, debido a que solo se esta resolviendo para una incógnita : presión); ***x*** es el vector columna que contiene las incógnitas (las presiones al tiempo *n+1*) y; ***b*** es el vector columna de términos independientes, conocido, al tiempo n. Nótese que debido a que las transmisibilidades son evaluadas al tiempo *n* (explícitamente) y las presiones son obtenidas al tiempo *n+1* se tiene un esquema semi-implícito.

De esta forma, por ejemplo, con una malla de 3 x 3 x 3, compuesta por 27 nodos, se tiene el sistema de la **figura 1.3**.

La Descomposición de Dominio, como se verá más adelante, se puede aplicar tanto en este sistema de ecuaciones (***Ax* = *b***) -dividiéndolo en subsistemas independientes- como en la malla de simulación -seleccionando las regiones que se desean analizar independientemente-.

$$
\begin{bmatrix}
c_1 & e_1 && n_1 &&&&&& u_1 \\
o_2 & c_2 & e_2 && n_2 &&&&&& u_2 \\
& o_3 & c_3 &&& n_3 &&&&&& u_3 \\
s_4 &&& c_4 & e_4 && n_4 &&&&&& u_4 \\
& s_5 && o_5 & c_5 & e_5 && n_5 &&&&&& u_5 \\
&& s_6 && o_6 & c_6 &&& n_6 &&&&&& u_6 \\
&&& s_7 &&& c_7 & e_7 &&&&&&&& u_7 \\
&&&& s_8 && o_8 & c_8 & e_8 &&&&&&&& u_8 \\
&&&&& s_9 && o_9 & c_9 &&&&&&&&& u_9 \\
l_{10} &&&&&&&&& c_{10} & e_{10} && n_{10} &&&&&& u_{10} \\
& l_{11} &&&&&&&& o_{11} & c_{11} & e_{11} && n_{11} &&&&&& u_{11} \\
&& l_{12} &&&&&&&& o_{12} & c_{12} &&& n_{12} &&&&&& u_{12} \\
&&& l_{13} &&&&&& s_{13} &&& c_{13} & e_{13} && n_{13} &&&&&& u_{13} \\
&&&& l_{14} &&&&&& s_{14} && o_{14} & c_{14} & e_{14} && n_{14} &&&&&& u_{14} \\
&&&&& l_{15} &&&&&& s_{15} && o_{15} & c_{15} &&& n_{15} &&&&&& u_{15} \\
&&&&&& l_{16} &&&&&& s_{16} &&& c_{16} & e_{16} &&&&&&&& u_{16} \\
&&&&&&& l_{17} &&&&&& s_{17} && o_{17} & c_{17} & e_{17} &&&&&&&& u_{17} \\
&&&&&&&& l_{18} &&&&&& s_{18} && o_{18} & c_{18} &&&&&&&&& u_{18} \\
&&&&&&&&& l_{19} &&&&&&&&& c_{19} & e_{19} && n_{19} \\
&&&&&&&&&& l_{20} &&&&&&&& o_{20} & c_{20} & e_{20} && n_{20} \\
&&&&&&&&&&& l_{21} &&&&&&&& o_{21} & c_{21} &&& n_{21} \\
&&&&&&&&&&&& l_{22} &&&&&& s_{22} &&& c_{22} & e_{22} && n_{22} \\
&&&&&&&&&&&&& l_{23} &&&&&& s_{23} && o_{23} & c_{23} & e_{23} && n_{23} \\
&&&&&&&&&&&&&& l_{24} &&&&&& s_{24} && o_{24} & c_{24} &&& n_{24} \\
&&&&&&&&&&&&&&& l_{25} &&&&&& s_{25} &&& c_{25} & e_{25} \\
&&&&&&&&&&&&&&&& l_{26} &&&&&& s_{26} && o_{26} & c_{26} & e_{26} \\
&&&&&&&&&&&&&&&&& l_{27} &&&&&& s_{27} && o_{27} & c_{27}
\end{bmatrix}^{n}
\begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \\ p_7 \\ p_8 \\ p_9 \\ p_{10} \\ p_{11} \\ p_{12} \\ p_{13} \\ p_{14} \\ p_{15} \\ p_{16} \\ p_{17} \\ p_{18} \\ p_{19} \\ p_{20} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{24} \\ p_{25} \\ p_{26} \\ p_{27} \end{bmatrix}^{n+1}
=
\begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \\ d_7 \\ d_8 \\ d_9 \\ d_{10} \\ d_{11} \\ d_{12} \\ d_{13} \\ d_{14} \\ d_{15} \\ d_{16} \\ d_{17} \\ d_{18} \\ d_{19} \\ d_{20} \\ d_{21} \\ d_{22} \\ d_{23} \\ d_{24} \\ d_{25} \\ d_{26} \\ d_{27} \end{bmatrix}^{n}
$$

Figura 1.3. Esquema de una matriz de coeficientes para una malla de 3 x 3 x 3.

CAPÍTULO II. DESCOMPOSICIÓN DE DOMINIO.

La solución numérica de problemas diferenciales de interés práctico pueden, con frecuencia, llevar a sistemas algebraicos de gran escala. Las modernas supercomputadoras hacen posible, hoy en día, afrontar un amplio rango de problemas que antes no podían afrontarse. Sin embargo, el tamaño de muchos de estos problemas es tan grande que se necesita una atención sustancial en el mejoramiento de los algoritmos numéricos existentes, así como, en el desarrollo de otros nuevos que se adapten mejor a la arquitectura de las supercomputadoras disponibles **(Quarteroni, A. y Valli, A., 1999)**.

Un esfuerzo importante que se está realizando, en este campo, es el de la Descomposición de Dominio (DD).

La primera técnica iterativa de Descomposición de Dominio conocida fue propuesta en el trabajo de H. A. Schwarz, en 1870, para probar la existencia de funciones armónicas en regiones irregulares que son la unión de regiones traslapadas. Variantes de este método fueron estudiadas después por Sobolev en 1936, Morgenstern en 1956 y Babuska en 1957 **(Chan, T. F. y Mathew, T. P., 1994)**. A pesar de todo esto, la Descomposición de Dominio para la solución numérica de ecuaciones diferenciales parciales es un campo nuevo (las primeras ideas importantes surgieron a inicios de los 80's).

En particular, la Descomposición de Dominio es una de las formas más significativas de crear algoritmos paralelos. Esto es importante, pues, las máquinas paralelas son necesarias para afrontar problemas numéricos a gran escala, como aquellos que surgen frecuente mente en muchas ramas de la física y de las ingenierías **(Quarteroni, A. y Valli, A., 1999)**.

Los métodos de Descomposición de Dominio son algoritmos paralelos, potencialmente rápidos y robustos para la solución iterativa de las ecuaciones, lineales o no lineales, que emergen de las discretizaciones de las ecuaciones diferenciales parciales.

Aunque estas técnicas pueden aplicarse directamente a las ecuaciones diferenciales parciales, son de mayor interés cuando se aplican sobre las discretizaciones de éstas (tanto para métodos de diferencias finitas como de elemento finito), tomando en cuenta que los sistemas, lineales o no lineales, que surgen de la discretización heredan muchas propiedades algebraicas de la ecuación diferencial parcial original **(Smith, B. F., [b])**.

La Descomposición de Dominio tiene como filosofía el viejo principio de "*Divide y Vencerás*". Y son técnicas empleadas para la solución de ecuaciones diferenciales parciales, resolviendo, iterativamente, subproblemas definidos en subdominios más pequeños que el dominio total. Es decir, se basan en una descomposición del dominio del problema en varios subdominios.

Las principales ventajas de los métodos de Descomposición de Dominio son:

1) Aumentan la capacidad de cómputo de las máquinas. Es decir, se pueden resolver problemas más grandes y con más detalle, que de otro modo sería imposible, debido al límite de memoria de las máquinas, o bien, al límite de los algoritmos de solución.
2) Pueden ser paralelizados. Es decir, pueden usarse con Programación en Paralelo para optimizar su tiempo de corrida al ser ejecutados por las máquinas. De esta forma, cada subdominio se resolverá simultáneamente en un procesador diferente.

2.1) TIPOS DE DESCOMPOSICIÓN DE DOMINIO:

El significado del término "Descomposición de Dominio" depende del contexto. Así, por ejemplo, puede referirse a técnicas óptimas de discretización, a solucionadores iterativos eficientes o, bien, a técnicas de paralelización. De hecho, en muchos códigos de simulación entran en juego diferentes aspectos de las técnicas de Descomposición de Dominio **(Wohlmuth, B. I., 2001)**.

En general, existen dos grandes clasificaciones de los métodos de Descomposición de Dominio; los que descomponen el dominio físico y los que descomponen el sistema de ecuaciones lineales.

Sin embargo, todo método de Descomposición de Dominio se basa en la suposición de que el dominio computacional dado, llamado Ω, es particionado en subdominios (Ω_i, $i=1,\ldots,m$), los cuales pueden estar traslapados o no. Y entonces, el problema original puede ser resuelto sobre cada subdominio Ω_i.

La **figura 2.1** muestra el caso en que los subdominios no están traslapados, sólo existe interfase. En esta figura, Ω_1 es el subdominio 1, Ω_2 es el subdominio 2 y B es la frontera que comparten los dos subdominios.

La **figura 2.2** muestra el caso en que los dominios están traslapados. Aquí, Ω_1 es el subdominio 1, Ω_2 es el subdominio 2, Γ_1 es la parte de la frontera de Ω_1 que está en el interior del traslape y, Γ_2 es la parte de la frontera de Ω_2 que está en el interior del traslape.

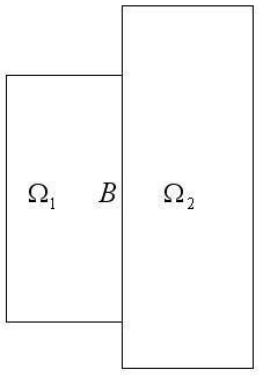


Figura 2.1. Subdominios sin traslape.

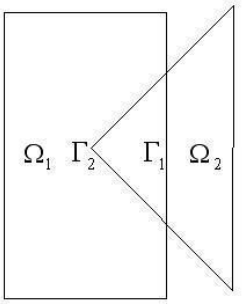


Figura 2.2. Subdominios con traslape.

2.1.1) Descomposición sobre el Dominio Físico:

Este tipo de descomposición es la más común y la más usada, por lo tanto, es la que más se encuentra en la literatura.

Consiste en dividir el dominio físico en varios subdominios, traslapados o no, los cuales son analizados en forma independiente y, después, unidos nuevamente para obtener la solución total del dominio.

El procedimiento es el siguiente:

1) Dividir el dominio en subdominios traslapados o no.
2) Resolver el problema en cada subdominio.
3) Formar la solución global juntando la solución de los subdominios.
4) Comparar los resultados en las zonas donde se comparte información (en el traslape –si lo hay- o en la interfase)

Si en el paso 4 la convergencia de los resultados no es satisfactoria se actualizan los datos y se repiten los pasos 2 al 4 hasta obtenerla. Si la convergencia es buena entonces se avanza en el tiempo, y se obtiene la solución global para un nuevo paso de tiempo.

Las **figuras 2.3** y **2.4** son ejemplos de descomposición sobre el dominio físico.

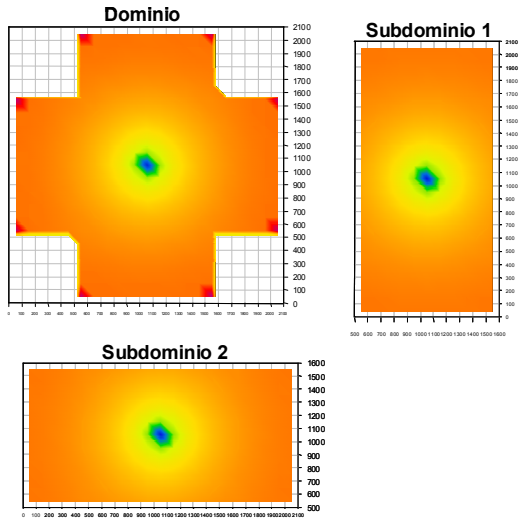


Figura 2.3. Descomposición del dominio físico.

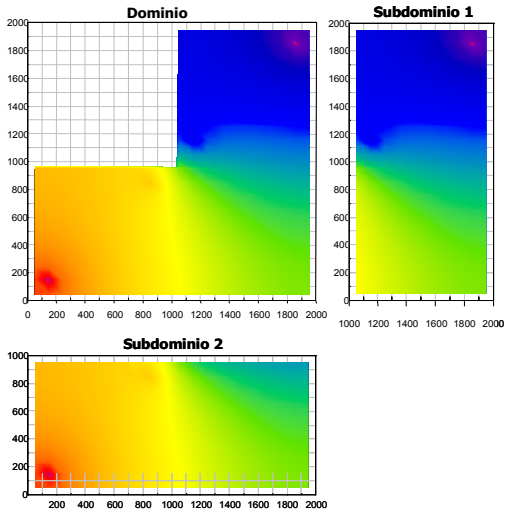


Figura 2.4. Descomposición del dominio físico.

La mayor dificultad de este tipo de descomposición radica en el trato que se les da a las condiciones de frontera internas y a los datos que caen sobre éstas; ver **figura 2.5.** Debe analizarse la forma en que se van a manejar y a actualizar los datos que están en las fronteras internas, donde los subdominios comparten información, pues, de esto depende la buena convergencia de los resultados. De hecho, entre los diferentes métodos de DD sobre el dominio físico, la diferencia principal está en la manera en que son tratadas las fronteras.

En el **Capítulo IV** de esta tesis se muestra la forma en que se trabajaron las condiciones de frontera internas al aplicar la descomposición del dominio físico sobre el simulador.

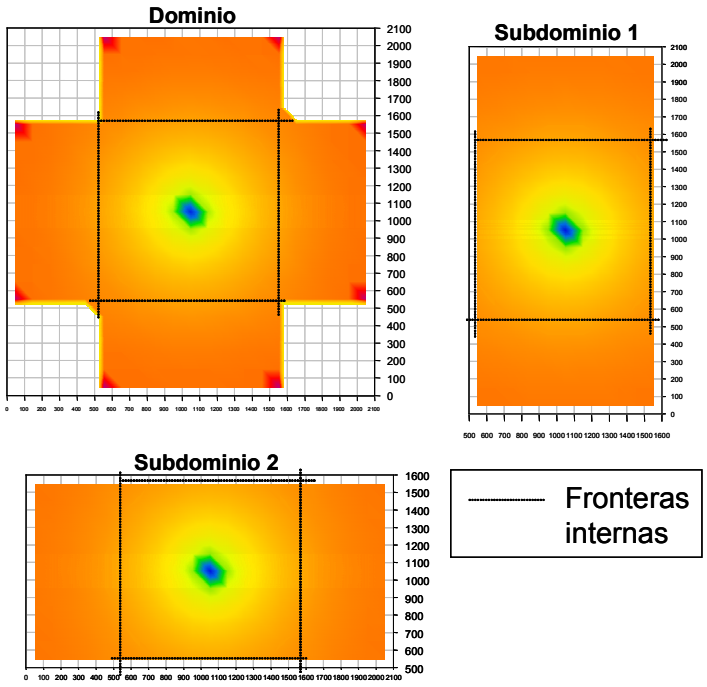


Figura 2.5. Fronteras internas en la descomposición del dominio físico.

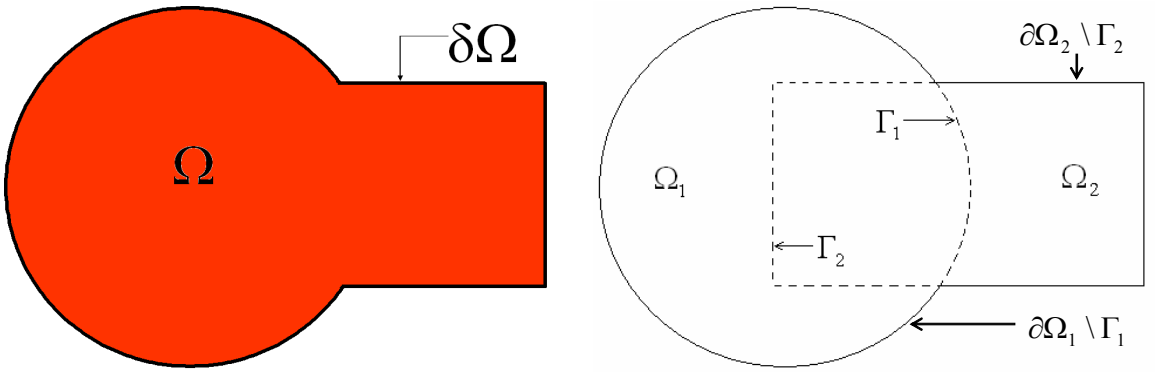
2.1.1.1) Descripción del método.

Como se dijo antes, el primer método de Descomposición de Dominio conocido fue el que dio a conocer Schwarz en 1870. Originalmente, el método no fue pensado como método numérico, sino, para resolver problemas elípticos con valores en la frontera sobre dominios que son la unión de dos regiones.

A continuación se da a conocer el Método Alternante se Schwarz, el cual es considerado como la base principal de los métodos de Descomposición de Dominio.

Método Alternante de Schwarz:

Considérese el dominio que se muestra en la **figura 2.6**, con $\Omega = \Omega_1 \cup \Omega_2$, como el dominio en el cual se desea resolver la ecuación diferencial parcial de la **ecuación 2.1.**



a) Dominio global y su frontera. **b)** Subdominios y sus fronteras.

Figura 2.6. Descomposición del dominio físico, representativa del método de Schwarz.

$$\boldsymbol{Lu} = \boldsymbol{f} \quad \text{en } \Omega . \qquad \textbf{…(2.1)}$$

Si se consideran los valores en la frontera se tiene:

$$\boldsymbol{u} = \boldsymbol{g} \quad \text{sobre } \partial\Omega .$$

La forma más simple de $\boldsymbol{L}$ es el Laplaciano (∇^2) el cual, en coordenadas cartesianas, está definido como:

$$\nabla \cdot \nabla = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$$

La frontera del dominio global, Ω, esta denotada por $\partial\Omega$. El interior del dominio global y de los subdominios está denotado por Ω y Ω_i, respectivamente. Es decir, Ω y Ω_i representan todo lo que hay dentro del dominio global y los subdominios sin considerar las fronteras de éstos. Se utiliza $\overline{\Omega}$ para denotar todo lo que está en el dominio global, tanto el interior (Ω) como las fronteras ($\partial\Omega$). Lo mismo sucede con los subdominios, se usa $\overline{\Omega}_i$ para representar todo lo que está dentro de ellos incluyendo sus fronteras. Por lo tanto, $\overline{\Omega}$ queda definido, en forma general, como $\overline{\Omega} = \Omega \cup \partial\Omega$. Las frontera artificiales o fronteras de los subdominios que están dentro del dominio global (fronteras internas) están definidas por Γ_i (para este caso *i=1,2*). El resto de las fronteras de los subdominios, las cuales no incluyen los puntos que están sobre Γ_i, están denotadas por $\partial\Omega_i \setminus \Gamma_i$.

Para describir el método alternante de Schwarz se definirán algunas notaciones más. Se usará $\boldsymbol{u_i^n}$ para denotar la solución aproximada sobre cada subsistema, $\overline{\Omega}_i$, después de *n* iteraciones y $\boldsymbol{u}_1^n|_{\Gamma_2}$ será una aproximación de $\boldsymbol{u}_1^n$ sobre Γ_2; de manera similar, $\boldsymbol{u}_2^n|_{\Gamma_1}$ será una aproximación de $\boldsymbol{u}_2^n$ sobre Γ_1. En otras palabras, $\boldsymbol{u}_1^n|_{\Gamma_2}$ es una aproximación que provee el subdominio 1, al ser resuelto, al subdominio 2 sobre su frontera Γ_2 y $\boldsymbol{u}_2^n|_{\Gamma_1}$ es una aproximación que provee el subdominio 2, al ser resuelto, al subdominio 1 sobre su frontera Γ_1.

El método alternante de Schwarz consiste en resolver en forma alternada los subdominios. Por ejemplo, se resuelve el subdominio 2 utilizando una aproximación inicial, $\boldsymbol{u}_2^0$. Luego, de su solución se obtiene una aproximación para el subdominio 1 sobre su frontera Γ_1, ya que ésta está dentro del subdominio 2, y con esta aproximación, $\boldsymbol{u}_2^n|_{\Gamma_1}$, y una aproximación inicial, $\boldsymbol{u}_1^0$, se resuelve el subdominio 1. De la solución del subdominio 1 se obtiene una nueva aproximación para el subdominio 2 sobre su frontera Γ_2, $\boldsymbol{u}_1^n|_{\Gamma_2}$, pues ésta está dentro del subdominio 1, y con ésta se resuelve el subdominio 2. Esto se realiza hasta que se alcanza la solución global deseada.

Esto se puede escribir como:

$$
\begin{aligned}
&\boldsymbol{L u_1^{\,n} = f_1} && \text{en } \Omega_1, \\
&\boldsymbol{u_1^{\,n} = g_1} && \text{sobre } \partial\Omega_1 \setminus \Gamma_1, \\
&\boldsymbol{u_1^{\,n} = u_2^{\,n-1}}|_{\Gamma_1} && \text{sobre } \Gamma_1.
\end{aligned}
\qquad \textbf{…(2.2)}
$$

En el subdominio 1. Y en el subdominio 2 como:

$$
\begin{aligned}
&\boldsymbol{L u_2^{\,n} = f_2} && \text{en } \Omega_2, \\
&\boldsymbol{u_2^{\,n} = g_2} && \text{sobre } \partial\Omega_2 \setminus \Gamma_2, \\
&\boldsymbol{u_2^{\,n} = u_1^{\,n}}|_{\Gamma_2} && \text{sobre } \Gamma_2.
\end{aligned}
\qquad \textbf{…(2.3)}
$$

De esta forma, en cada parte del paso del método alternante de Schwarz se resuelve un problema con valor en la frontera sobre el subdominio Ω_i con los valores de frontera dados, $\boldsymbol{g}$, sobre la frontera, $\partial\Omega_i \setminus \Gamma_i$, y con la solución aproximada previa sobre la frontera interior Γ_i **(Smith, B. F., Bjørstad, P. E. y Gropp, W., 1996)**.

Las ventajas que brinda la descomposición sobre el dominio físico:

- Se pueden atacar problemas con geometría compleja, pues, el dominio es descompuesto en subdominios con geometría más sencilla.
- Se pueden afrontar problemas de heterogeneidad, al dividir en subdominios con características especiales y diferentes a las de los demás. Así, por ejemplo, la malla de un yacimiento heterogéneo puede dividirse en tantos subdominios como zonas homogéneas tenga.
- Se pueden resolver dominios más grandes, ya que, al descomponerlos en subdominios independientes de menor dimensión, la carga de cómputo disminuye y, de esta forma, no se excede la capacidad tanto del hardware como de los algoritmos.
- Se pueden resolver problemas en forma más detallada, por ejemplo; trabajar con una malla más fina. Esto se puede hacer, ya sea, utilizando una malla más fina en todos los subdominios, o bien, en algunos de éstos, como se muestra en la **figura 2.7**.

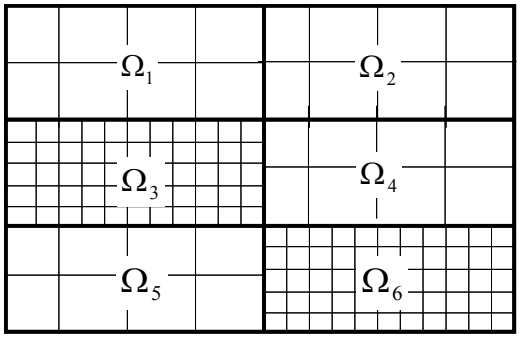


Figura 2.7. Refinamiento de la malla de simulación.

2.1.2) Descomposición sobre el sistema de ecuaciones lineales:

A este tipo de descomposición también se le llamará, en este trabajo, descomposición sobre el dominio matemático.

En este tipo de descomposición se divide el sistema de ecuaciones lineales que define al problema en subsistemas. Es decir, ya no se trabaja sobre la región física, sino, sobre el sistema de ecuaciones lineales que representa el fenómeno estudiado y su región física y, gracias a esto, las fronteras se manejan implícitamente en el sistema de ecuaciones que representa al problema original completo.

El trabajo realizado en esta tesis está basado, principalmente, en este tipo de descomposición, en el **Capítulo IV** se exponen las razones de esto.

El procedimiento general consiste en:

1) Dado el sistema global $Ax=b$, descomponerlo en subsistemas $A_i x_i = b_i$, traslapados o no.
2) Resolver cada subsistema en forma independiente.
3) Formar la solución global juntando la solución de los subsistemas, $x = x_1 \cup x_2 \cup, ..., \cup x_m$.

4) Revisar que la solución global cumpla con cierta tolerancia.

Si en el paso 4 no se cumple con la tolerancia se actualizan los datos y se repiten los pasos 2 al 4 hasta satisfacerla. Si se cumple con la tolerancia se avanza en el tiempo y se resuelve el sistema global para un nuevo paso de tiempo.

2.1.2.1) Descripción del método.

Se da a conocer el Método Multiplicativo de Schwarz debido a que es el método que se utilizó para dividir el sistema de ecuaciones lineales.

Método Multiplicativo de Schwarz:

Considérese que se tiene una ecuación diferencial parcial de la forma:

$$Lu = f \qquad \textbf{...(2.4)}$$

Debido a que esta ecuación es muy difícil de resolver en forma analítica, se discretiza para ser resuelta numéricamente y se obtiene un sistema lineal de la forma:

$$Ax^{n+1} = b \qquad \textbf{...(2.5)}$$

Donde $\boldsymbol{A}$ es la matriz de coeficientes que representa al sistema, $\boldsymbol{x}^{n+1}$ representa el vector de incógnitas al tiempo $n+1$, porque se está resolviendo en forma implícita, y $\boldsymbol{b}$ es el vector de términos independientes. En lo posterior el vector de incógnitas $\boldsymbol{x}^{n+1}$ se representará sólo como x para simplificar la notación y facilitar la comprensión.

Este sistema puede ser resuelto de forma directa como:

$$x = A^{-1}b \qquad \textbf{...(2.6)}$$

En la Simulación Numérica de Yacimientos el **sistema (2.5)** tiene la forma que se muestra en la **figura 2.8.** Además, como se puede ver, es similar al mostrado en la **figura 1.3**, sólo que, las siete bandas de la matriz no se pueden apreciar debido a la dimensión del sistema y a la escala manejada en su representación.

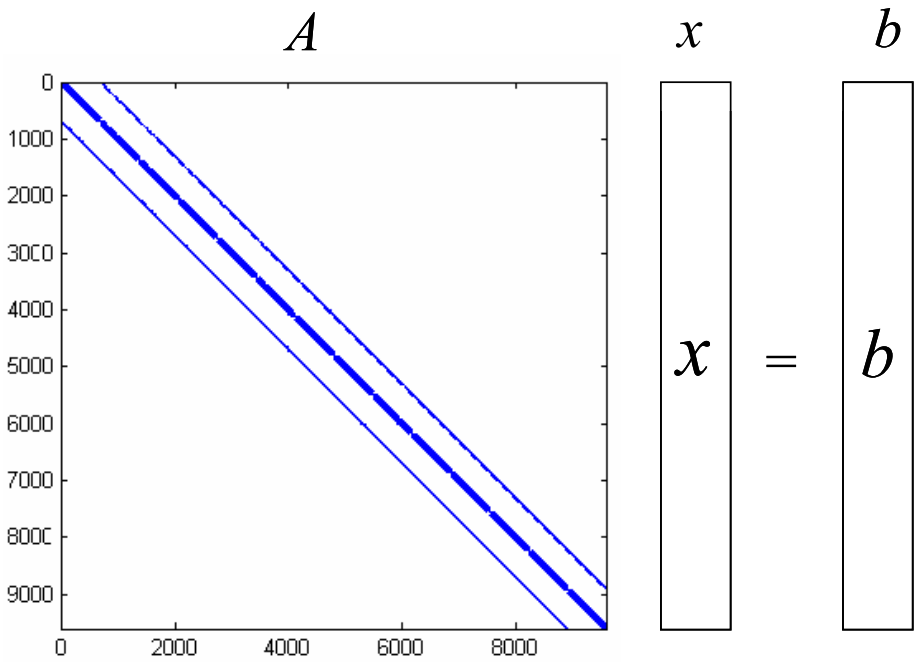


Figura 2.8. Esquema matricial del sistema 2.5

Al aplicar la descomposición sobre el sistema de ecuaciones **(2.5)**, considerando que es subdividido en dos, y aprovechando la forma de la matriz de coeficientes se obtienen subsistemas más pequeños, tal como se muestra en la **figura 2.9.**

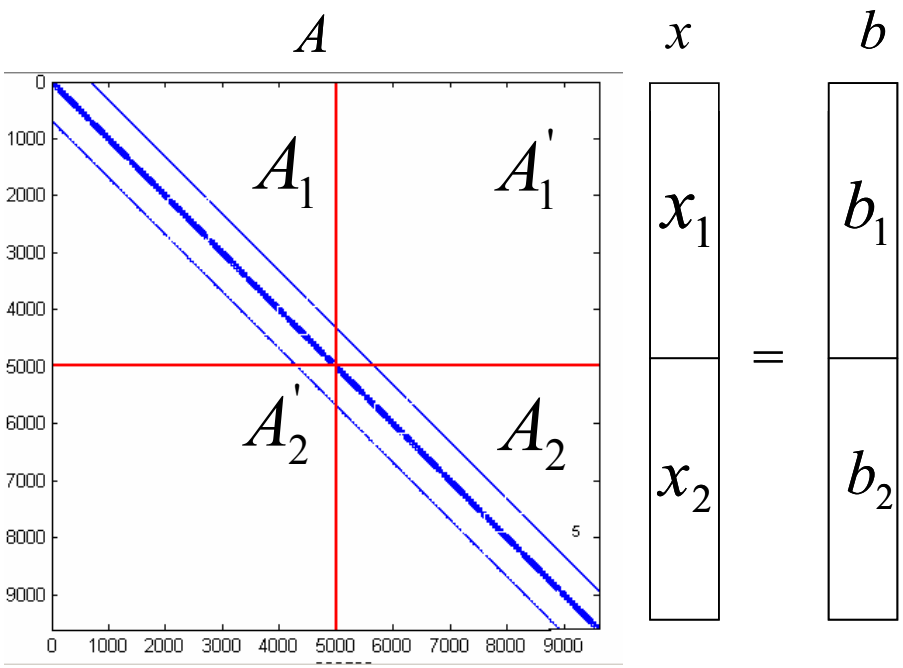


Figura 2.9. Descomposición del sistema de ecuaciones.

Los subsistemas obtenidos tienen la forma:

$$A_1x_1 + A_1'x_2 = b_1$$
$$A_2'x_1 + A_2x_2 = b_2 \qquad \text{…(2.7)}$$

Pero, como se puede observar en la **figura 2.9**, A_1' y A_2' tienen una contribución mínima sobre la solución del sistema, debido a que la mayoría de sus elementos son ceros; despreciando estos términos, los subsistemas se reducen a:

$$A_1x_1 = b_1$$
$$A_2x_2 = b_2 \qquad \text{…(2.8)}$$

En forma general, para m subdominios, la **ecuación 2.8** se puede escribir:

$$A_jx_j = b_j \qquad \text{…(2.9)}$$

Donde: $j = 1, 2, \ldots, m$, indica el subsistema correspondiente.

Al considerar que el sistema global es dividido en los dos subsistemas de la **ecuación 2.8**, ahora independientes, se recomienda no utilizar un método de solución directa, por lo que se recurre a una solución iterativa. Además, al despreciar los términos A_1' y A_2' se esta generando un error. Este error puede ser reducido considerando una región de traslape entre los subdominios o considerando los términos en $A_1'x_2$ y $A_2'x_1$ diferentes de cero en el lado derecho de cada subsistema al tiempo n.

La solución iterativa del sistema es la siguiente:

1) Se asignan las condiciones iniciales; la matriz de coeficientes A es llenada, x es el vector solución y b tiene las variables independientes.
2) Se divide el sistema en m subsistemas, como se muestra en la **ecuación 2.9**.
3) Se toma una aproximación global inicial, x_o^i. Al dividir en subsistemas a cada uno de éstos le corresponde una parte de la aproximación inicial. Entonces, se tiene:

$$x_0^{\,i} = x_1^i \cup x_2^i \cup \ldots \cup x_m^i \qquad \textbf{…(2.10)}$$

Donde; i indica el número de iteración para un mismo paso de tiempo. Es decir, no se podrá avanzar en el tiempo hasta que se alcance la convergencia.

Nótese que los componentes que forman a x_o^i no se están sumando sino que están unidos. Esto se debe a que, como se puede apreciar en la **figura 2.9,** los vectores x_1, x_2 , ..., x_m son partes del vector total x, así que, para formar al vector x_0^i sólo deben unirse.

4) Cada uno de los subsistemas es resuelto en forma independiente obteniendo así una parte de la solución global. Para la solución de los subsistemas cualquier método de solución de sistemas de ecuaciones lineales puede ser aplicado.
5) La siguiente aproximación de la solución global es conformada por la solución de cada uno de los subsistemas, como se muestra en la **ecuación 2.11**.

$$x_0^{\,i+1} = x_1^{i+1} \cup x_2^{i+1} \cup, \ldots, \cup x_m^{i+1} \qquad \textbf{…(2.11)}$$

6) Se considera que la solución es obtenida cuando la diferencia entre la aproximación actual y la anterior cumpla con una tolerancia, definida por:

$|x^{i-1}-x^{i}| \leq$ tolerancia **…(2.12)**

Los pasos 3) al 6) son repetidos hasta cumplir la condición del paso 6) en cada paso de tiempo.

Ventajas de la descomposición sobre el sistema de ecuaciones lineales:

- Se puede trabajar casi con cualquier clase de problema, sin importar su dominio físico o malla, pues, se trabaja sólo con el sistema de ecuaciones lineales que representa al problema.
- Se pueden resolver problemas más grandes o más detallados, ya que, al descomponer su sistema de ecuaciones en subsistemas independientes de menor tamaño, la carga de cómputo disminuye.
- Se puede utilizar cualquier método de solución de ecuaciones lineales para resolver los subsistemas, pues, éstos son independientes.

CAPÍTULO III. PROGRAMACIÓN EN PARALELO.

La Programación en Paralelo ha nacido como un esfuerzo por incrementar las capacidades de cómputo de las máquinas, tanto en su rapidez de procesamiento como en su capacidad para manejar grandes cantidades de datos.

Los resultados obtenidos han sido buenos y alentadores, por eso, su uso se ha incrementado y ha llamado la atención no sólo de los científicos del área de la computación, sino, de todas las ramas de la Ciencia y de las ingenierías. Debido a esto, se han realizado fuertes investigaciones e inversiones en este campo, sobre todo a partir de finales de los 80's.

Además de todo esto, el uso cada vez más frecuente de la Descomposición de Dominio en la simulación numérica, ha contribuido fuertemente en la aplicación de la Programación en Paralelo, debido a que, por sus características, esta herramienta tiene su mejor desempeño con la Programación en Paralelo.

Por otra parte, en los últimos años una gran variedad de máquinas paralelas se han vuelto disponibles y, éstas son capaces de alcanzar el rendimiento de las supercomputadoras si sus recursos son usados eficientemente **(Cheng, W-S, Jameson, A. y Mitty, T. J., 1995)**. Esto es de gran importancia, pues, el costo de una máquina paralela es mucho menor que el de una supercomputadora. De ahí que, el uso efectivo de estas máquinas sea tan atractivo.

En cuanto a la Programación en Paralelo, ésta consiste en la división de un problema, presentado como una serie de datos o serie de acciones, entre múltiples elementos de procesamiento que operan simultáneamente **(Ortega A., J. L., 2003)**. En otras palabras, consiste en la coordinación de recursos computacionales que trabajan simultáneamente para alcanzar un objetivo común **(Ortega A., J. L., 1998)**.

La filosofía de la Programación en Paralelo se basa en el principio de "*la Unión hace la Fuerza*", ya que, en vez de usar un solo procesador se usan varios para atacar un problema.

El poder del paralelismo descansa en la partición de un problema grande en relación a su complejidad o carga de trabajo. Esta partición es necesaria para dividir este gran problema en subproblemas más pequeños, más fáciles de comprender y que se pueden trabajar por separado en una forma más cómoda. Esto es un factor clave en los sistemas paralelos, ya que, la partición permite no sólo crear por separado los componentes de software, sino, también, ejecutarlos simultáneamente **(Ortega A., J. L., 2003)**.

Las principales ventajas de la Programación en Paralelo son:

- Realizar tareas a una escala que para otros sistemas de cómputo sería imposible o no rentable.
- Optimizar el tiempo de ejecución de los programas, especialmente si son robustos.

3.1) Razones para el uso de la Programación en Paralelo

El paralelismo es un concepto atractivo e intuitivo. Considérese un problema de ciencia computacional o de ingeniería en el cual se está trabajando. Si al ejecutarse en un solo procesador se obtienen los resultados en 10 horas, ¿Por qué no usar 10 procesadores y obtener los resultados después de sólo una hora?.

En teoría, el paralelismo es así de sencillo; aplicar muchos procesadores a un solo problema. Para los científicos de la computación esta es una buena forma de vencer las restricciones que existen en las máquinas de un solo procesador. Además, aparte de ofrecer soluciones más rápidas, las aplicaciones que han sido paralelizadas (convertidas a programas paralelos) pueden resolver problemas más grandes y más complejos, los cuales con sus datos de entrada o resultados intermedios exceden la capacidad de memoria de un solo procesador. De esta forma, las simulaciones pueden realizarse con una mayor resolución y los fenómenos físicos pueden modelarse de una manera más real.
En la práctica, sin embargo, el paralelismo implica un gran esfuerzo. La Programación en Paralelo involucra una curva de aprendizaje muy inclinada. El programador debe pensar de

su aplicación en nuevas maneras y puede que termine rescribiendo casi todo el código serial. Por otro lado, las técnicas para depurar fallos y afinar el desempeño de los programas seriales no se extienden fácilmente en lo paralelo. Es muy posible que se trabaje durante meses paralelizando una aplicación sólo para encontrar que brinda resultados incorrectos o que corre más lento que antes.

3.2) Evaluación del esfuerzo al paralelizar un simulador

El propósito y la naturaleza del simulador son los indicadores más importantes de qué tan exitosa será la paralelización. La elección de la máquina paralela y el plan de ataque también tendrán un impacto significativo, no sólo en el desempeño, sino, en el nivel de esfuerzo requerido para la paralelización.

A continuación se mencionan algunos aspectos que ayudan a saber si vale la pena paralelizar un programa o no.

Precondiciones para el paralelismo:

Básicamente, el propósito de la aplicación es un buen indicador de cuánto se puede invertir en mejorar su rendimiento. Tres factores establecen los objetivos de desempeño de una aplicación.

El Primer factor se basa en qué tan frecuente será usada la aplicación antes de que sean necesarios algunos cambios. Si la respuesta es miles de veces entre revisiones, entonces, esta es una aplicación que probablemente merece un esfuerzo para mejorar su desempeño. En cambio, un programa que necesita ser modificado frecuentemente no permitirá amortizar el tiempo invertido en las mejoras.

El segundo factor es el tiempo actualmente necesario para ejecutar la aplicación. Si por ejemplo, se necesitan días para obtener los resultados. Reduciendo este tiempo a una

fracción se puede optimizar fuertemente la productividad. En contraste, si el tiempo de corrida es de unos cuantos minutos, las ganancias obtenidas en el rendimiento contra el esfuerzo requerido en la paralelización serán muy bajas. Sin embargo, estas mediciones son relativas, ya que, si la aplicación que se desea mejorar es un sistema de administración de emergencia en tiempo real, incluso una optimización de segundos puede muy buena.

El tercer factor indica la satisfacción que se tiene con la complejidad y la resolución de los resultados. Si la rapidez o la capacidad de memoria de las máquinas seriales restringen el simulador, por ejemplo, obligando al uso mallas gruesas en vez de mallas finas, la paralelización puede ser una forma de vencer estas restricciones.

La **figura 3.1** muestra un espectro en el cual caen los tres objetivos y en donde se puede ver qué se puede ganar con la paralelización.

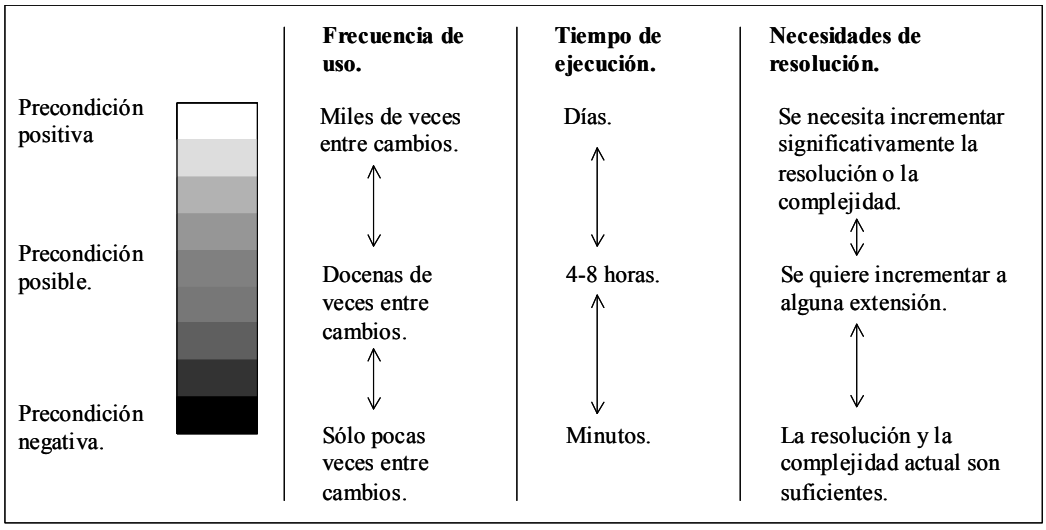


Figura 3.1. Precondicionamiento de paralelización. **(Pancake, Ch. M., 1996)**

De acuerdo a la experiencia, de algunos científicos e ingenieros, las necesidades deben alcanzar por lo menos un color blanco, del espectro de la **figura 3.1**, antes de considerar invertir esfuerzo en paralelizar una aplicación. Pero, por otro lado, un solo color negro puede interpretarse como una indicación de que el rendimiento no amerita mucho esfuerzo de paralelización. Además, incluso tres blancos no garantizan que el paralelismo dará

buenos resultados, simplemente indican que la aplicación necesita poder potencial de paralelización **(Pancake, Ch. M., 1996)**.

3.3) Influencia del problema sobre el rendimiento. Tipos de paralelismo.

La naturaleza del problema es una pieza clave en el éxito o fracaso de la programación en paralelo. Geoffrey Fox fue el primero en investigar cómo las características de la aplicación restringen el desempeño. Él estableció que muchas aplicaciones técnicas caen en una de tres categorías, que llamó *problema de arquitecturas*, y que es satisfecha con ciertos tipos de computadoras paralelas. Pero, **(Pancake, Ch. M., 1996)** extiende el concepto a cuatro categorías, introduciendo el *paralelismo de tubería*. A continuación de se da a conocer su clasificación.

Considérese un problema de mapeo sísmico. Los datos de la respuesta al choque sísmico son recolectados en diferentes sitios en el campo, luego son trabajados por computadora para construir curvas de contorno de la estructura geológica subsuperficial de cada sitio. El trabajo de cómputo puede ser una secuencia de trabajos seriales, donde en cada uno se construye una imagen con cada serie de datos; o se puede usar el paralelismo procesando múltiples series de datos al mismo tiempo. Como se muestra en la **figura 3.2.**

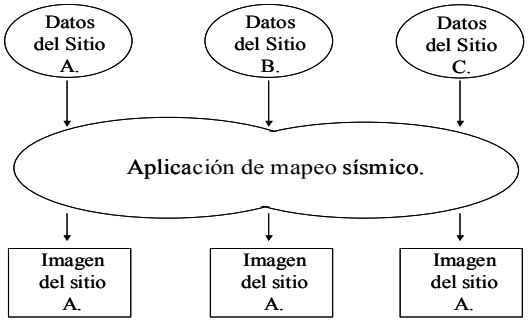


Figura 3.2. Esquema de paralelismo perfecto. **(Pancake, Ch. M., 1996)**

Este es el estilo de problema más sencillo, referido como *paralelismo perfecto*. Los cálculos en cada serie de datos son totalmente independientes. Es decir, las imágenes pueden ser procesadas en máquinas independientes que tengan copia de la aplicación. Es fácil mejorar el desempeño significativamente en aplicaciones que tienen este tipo de paralelismo.

Ahora supóngase que las imágenes no son totalmente independientes, talvez, las respuestas de la subestructura están siendo simuladas en series de pasos de tiempo, como se muestra en la **figura 3.3**, los datos de diferentes pasos de tiempo son usados para generar imágenes que muestren el cambio con el tiempo. Los datos producidos en la simulación deben utilizarse para generar un volumen tridimensional y luego recibir un cierto formato para ser visualizados gráficamente. Si esta aplicación fuera ejecutada serialmente, las series de datos de salida del simulador pueden servir como datos de entrada para el programa generador del volumen tridimensional, del cual sus datos de salida servirán como datos de entrada para el programa de visualización gráfica. El paralelismo, en este problema, puede ser introducido de tal forma que, la generación del volumen tridimensional comience tan pronto como estén disponibles los datos de salida del primer paso de tiempo. Luego, mientras el simulador produce la tercera serie de datos, el programa de generación de volumen trabaja con la segunda serie de datos, y la primera serie de datos es formateada y mostrada gráficamente.

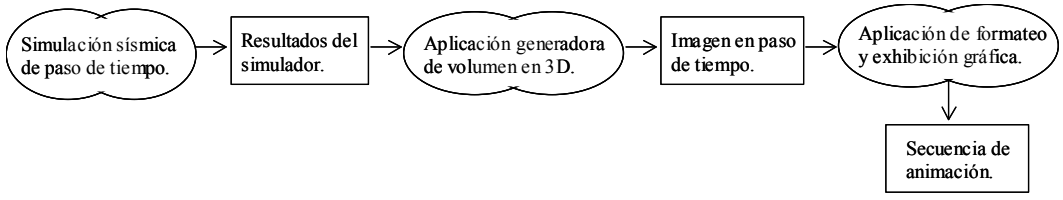


Figura 3.3. Esquema de paralelismo de tubería. **(Pancake, Ch. M., 1996)**

Este modelo es llamado *paralelismo de tubería*, ya que los datos siguen una secuencia como de "tubería" al pasar de una etapa computacional a otra. En este modelo los resultados son pasados de una sola manera a través de la tubería, es decir; la simulación del

siguiente paso de tiempo no requiere información de la generación del volumen en 3D o de la visualización gráfica. El inicio de todas las secuencias es comenzado conforme los datos de entrada se vuelven disponibles, de tal forma que el mejoramiento del desempeño global dependerá del número de pasos de tiempo que serán procesados una vez que todos los puntos de la tubería están activos. Este tipo de paralelismo implica algunos problemas. Si las etapas no son computacionalmente equivalentes, las más rápidas saturarán a las más lentas, acabando primero. Una manera de solucionar esto, consiste en ejecutar etapas computacionalmente intensivas en procesadores rápidos, pero, balancear el trabajo puede ser una tarea muy difícil. Por esta razón, el paralelismo de tubería no es tan simple como el paralelismo perfecto.

En muchas aplicaciones los resultados no pueden estar restringidos al flujo en una sola dirección entre las etapas de procesamiento, como en el paralelismo de tubería. Por ejemplo, considérese un problema de dinámica atmosférica. Los datos representan un modelo tridimensional de la atmósfera, donde la ocurrencia en una región tiene influencia sobre áreas subyacentes y suprayacentes al disturbio y, talvez, en zonas vecinas horizontalmente. Con el tiempo, los efectos se propagan en todas direcciones cubriendo un área mayor; incluso la fuente del disturbio puede experimentar resonancias u otros movimientos de regiones vecinas. Si esta aplicación fuera ejecutada en forma serial, los cálculos serían realizados en todos los datos para obtener un estado atmosférico promedio, luego, una nueva iteración comenzaría. En este problema, la Programación en Paralelo es introducida con varios procesadores participando en una iteración, en la cual cada uno aplica los cálculos a un grupo de la serie global de datos. Ver **figura 3.4.** Cada iteración es concluida en todos los procesadores antes de que una nueva iteración comience.

Esto es llamado *paralelismo totalmente sincrónico*. En este modelo de paralelismo, cada cálculo es aplicado sincrónicamente (o simultáneamente) a todos los datos. La clave de esto es que los cálculos o las decisiones futuras dependen de los resultados de todos los cálculos anteriores. Usualmente no hay tantos procesadores como para trabajar al mismo tiempo todos los grupos de datos, así que cada procesador trabaja con más de un grupo de datos. Si los grupos de datos no son homogéneos, la intensidad del trabajo computacional variará en

los diferentes procesadores. Por ejemplo, un disturbio en el estrato que está hasta arriba comienza modificando los datos de los estratos superiores, pues son los más cercanos, mientras que los estratos inferiores permanecen sin cambio. Esta variación espacial significa que si cada procesador realiza los cálculos a un grupo de datos en forma horizontal, sólo uno o dos procesadores estarían al inicio realizando un trabajo intensivo. Mientras tanto, la sincronización impide que los otros procesadores puedan avanzar y realizar los siguientes cálculos hasta que los procesadores ocupados terminen. Por otro lado, si los procesadores aplican los cálculos a regiones verticales, el trabajo puede estar uniformemente distribuido al inicio de la corrida, pero, esto cambiaría en los siguientes pasos de tiempo, cuando los cálculos difieran el la dirección horizontal. Consecuentemente, el paralelismo totalmente sincrónico requiere más esfuerzo del programador que el paralelismo de tubería para alcanzar buenos resultados.

El cuarto estilo de paralelismo es ilustrado con una aplicación relacionada, la cual modela la difusión de contaminantes a través del agua subterránea. Ver **figura 3.5.** Inicialmente, sólo las particiones de agua subterránea cercanas a la fuente de contaminación son afectadas, pero conforme avanza el tiempo los contaminantes se esparcen, formando áreas irregulares de concentración.

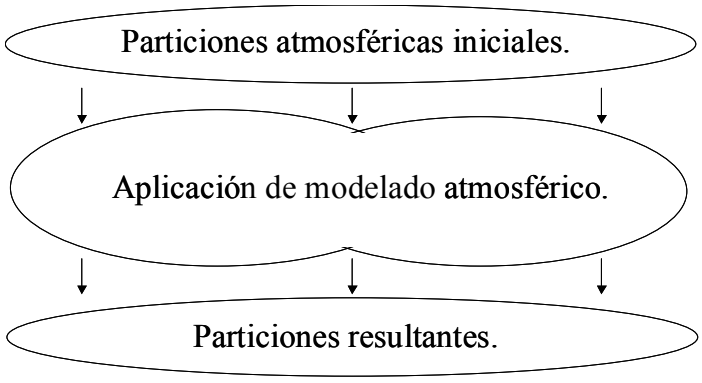


Figura 3.4. Esquema de paralelismo totalmente sincrónico. **(Pancake, Ch. M., 1996)**

La carga de computación depende de la cantidad de contaminante y de la estructura geofísica, así que varía dramáticamente de una partición (y paso de tiempo) a otra. En un

programa serial esto significa que la longitud del paso de tiempo es irregular y, talvez, impredecible. El paralelismo es aplicado dividiendo el trabajo a lo largo de muchos procesadores a cada paso de tiempo. Durante los primeros pasos de tiempo cada procesador puede aplicar cálculos a sólo unas cuántas particiones, y la duración de los cálculos será breve debido a que las concentraciones son bajas; pero, después, conforme las concentraciones van creciendo y afectando otras particiones, un solo procesador puede realizar muchos más cálculos en muchas más particiones a cada paso.

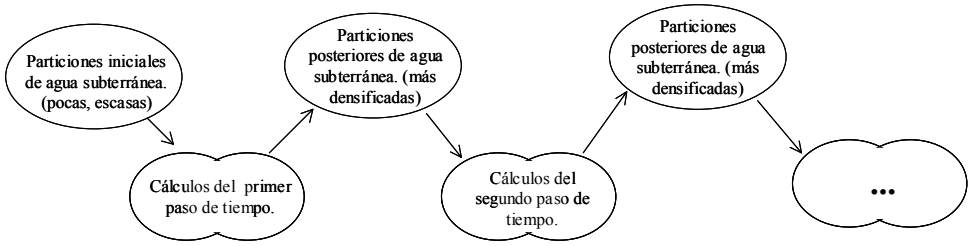


Figura 3.5. Esquema de paralelismo pobremente sincrónico. **(Pancake Ch. M. 1996)**

Este es el *paralelismo pobremente sincrónico*. Cuando cada paso de tiempo termina, los procesadores que han terminado su trabajo deben esperar a que los demás terminen antes de compartir resultados intermedios y comenzar el siguiente paso de tiempo. De esta forma, la característica clave de este estilo es que cada procesador realiza una parte del problema, intercambiando información intermitentemente. Este tipo de paralelismo es el más difícil de programar. La necesidad de intercambiar información entre los procesadores requiere de pruebas para que un procesador pueda determinar cuándo los datos de los demás procesadores están listos y pueda evitar sobrescribir valores que no se han usado aún. Con este tipo de paralelismo es muy difícil distribuir la carga de trabajo entre los procesadores, ya que, ésta varía temporal y espacialmente.

La clasificación anterior es propuesta por **(Pancake, Ch. M., 1996)**, pero, existen muchas otras. Para mayor información, se recomienda revizar: **(Gropp, W., Lusk, E. y Skjellum, A.)**, **(Jean-Yves, B. y Laurent, C., 1997)** y **(Ortega A., J. L., 2000)**.

Analizar el problema de arquitectura ayuda a decidir si el paralelismo vale la pena o no. Primero, se debe considerar cómo usa los datos la aplicación, luego, se identifica el tipo de paralelización que se necesita para la aplicación y, después, se determina cómo influirá esto en el esfuerzo de paralelización aplicando las siguientes reglas de dedo **(Pancake, Ch. M., 1996)**:

1) Si la aplicación cumple con el modelo de paralelización perfecta, la tarea de paralelización es relativamente fácil y se puede alcanzar un buen desempeño.
2) Si la aplicación cumple con la paralelización de tubería, se tiene que realizar un mayor trabajo; pero, si no se puede balancear la carga de cómputo puede que no valga la pena el esfuerzo.
3) Si la aplicación es totalmente sincrónica, una cantidad significativa de esfuerzo es requerida y los beneficios serán mínimos; la decisión de paralelización debe basarse en qué tan uniforme será la carga de cómputo.
4) Si la aplicación es pobremente sincrónica significa que la paralelización será muy difícil y probablemente no valga la pena el esfuerzo, a menos que las interacciones entre los procesadores sean mínimas (poco frecuentes).

3.4) Influencia de la maquina en el desempeño del programa paralelo

Una computadora paralela es, generalmente, cualquier colección de elementos de procesamiento conectados por un tipo de red de comunicación. El término elementos de procesamiento se refiere principalmente a procesadores, pero, también involucra a la memoria. También son conocidas como multicomputadoras. El rendimiento de una aplicación paralela depende de la rapidez y capacidad de los procesadores, así como, de la comunicación entre ellos.

La **figura 3.6** muestra un "árbol de familia" básico para arquitecturas de computadoras paralelas. El *modelo de control* establece cuántas diferentes instrucciones pueden ejecutarse simultáneamente. Los términos SIMD (Single Instruction, Múltiple Data) y MIMD

(Múltiple Instruction, Múltiple Data) existen desde los primeros tiempos de la Programación en Paralelo y ambas son, en esencia, el único factor distintivo entre las diferentes máquinas en paralelo. El *modelo de memoria* indica cuántos procesadores pueden acceder directamente a una locación de memoria dada. En las computadoras de memoria compartida todos los procesadores acceden a una sola memoria, mientras que las computadoras de memoria distribuida usan una memoria por separado para cada procesador. La memoria es compartida entre pequeños grupos de procesadores en computadoras de multiprocesadores simétricas (SMP), pero, cuando son agrupados los grupos para formar sistemas más grandes, la memoria de cada grupo permanece aislada. El *modelo de programación* se refiere a restricciones en el número de ejecutables que pueden participar en una ejecución paralela. En el modelo MPMD todas las instrucciones que serán llevadas a cabo por todos los procesadores son combinadas dentro de un solo ejecutable.

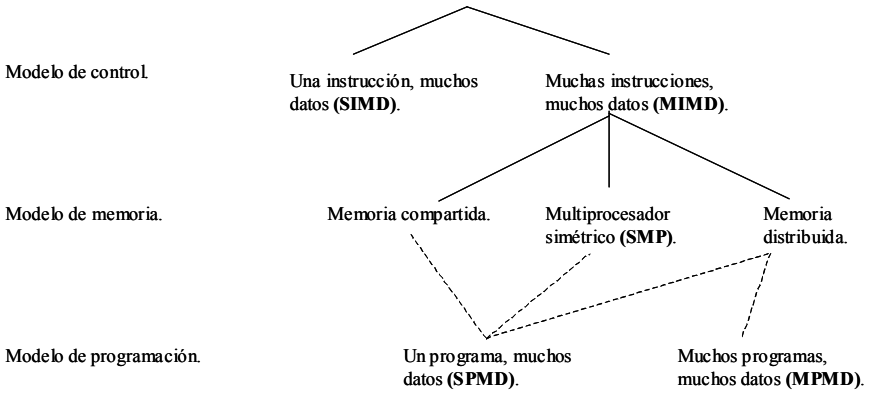


Figura 3.6. Genealogía de sistemas de cómputo paralelos. **(Pancake, Ch. M., 1996)**

La interacción del modelo de control y el modelo de memoria deriva en cuatro clases de arquitecturas de computadoras paralelas; SIMD, de memoria compartida, de memoria distribuida y SMP. Éstas son descritas, en forma general, a continuación.

La **tabla 3.1** muestra las principales características de estos cuatro tipos de computadoras paralelas.

3.4.1) Multicomputadoras SIMD:

En las multicomputadoras SIMD todos los procesadores ejecutan la misma instrucción.

En este tipo de máquinas, la unidad de control es la clave principal para obtener los beneficios de la paralelización. Las máquinas SIMD son relativamente fáciles de programar y usan la memoria eficientemente. El principal obstáculo de programación consiste en llevar a cabo cálculos básicos a través de operaciones de arreglo (operaciones de vector). Si la aplicación no cumple con el modelo de paralelización totalmente sincrónica, la tarea de paralelización en una SIMD será muy difícil o imposible.

Algunos ejemplos de estas máquinas son; la MP-2 de la MasPar y la Connection Machine de la Thinking Machines.

3.4.2) Multicomputadoras de Memoria Compartida:

Distintas a las máquinas SIMD, las máquinas MIMD dan a cada procesador su propia unidad de control. A cualquier momento durante la ejecución, los diferentes procesadores pueden ejecutar diferentes instrucciones.

En una multicomputadora de memoria compartida, los procesadores interactúan accediendo a locaciones de memoria de una memoria compartida. Las máquinas MIMD tienden a ser las computadoras más rápidas, grandes y caras que existen. Y, además de que son más difíciles de programar que las máquinas SIMD, ofrecen un mejor ajuste a un gran rango de aplicaciones.

Algunos ejemplos de este tipo de máquinas son; la Cray Y/MPs y la Fujitsu VPs.

3.4.3) Multicomputadoras de Memoria Distribuida:

En las máquinas de memoria distribuida cada procesador ejecuta su propia corrida, pero, como su nombre lo indica, cada procesador tiene su memoria privada. Muchas de las computadoras paralelas de alto rendimiento actuales son de este tipo. Algunos ejemplos son; la Cray T3D, IBM SP-2, Intel Paragon y Meiko CS-2. Basados en la tecnología de microprocesador de estación de trabajo, estos sistemas son versátiles y rentables.

Las paralelizaciones de tubería y pobremente sincrónicas pueden alcanzar un rendimiento respetable en este tipo de máquina sólo si hay pocos datos a ser intercambiados o periodos relativamente largos para efectuar los intercambios. Las aplicaciones totalmente sincrónicas no son aptas para este tipo de sistema.

3.4.4) SMPs y Clusters SMP:

Conocidas como máquinas multiprocesadoras simétricas, son nuevas en el mercado de máquinas paralelas. La palabra “simétrica” se refiere al hecho de que cada procesador puede recuperar datos almacenados en cualquier locación de memoria en la misma cantidad de tiempo. Las SMPs se parecen a las multicomputadoras de memoria compartida, pero, son más lentas, menos caras y con menor poder de procesador. Algunos ejemplos son; la PowerChallenge de SGI y la Sparcserver de Sun.

Es posible unir SMPs en grandes grupos (cluster SMP) y, así, darles un poder de procesamiento mayor. La configuración resultante se comporta como una máquina de memoria distribuida, a excepción de que cada nodo tiene múltiples procesadores compartiendo una memoria común.

Tabla 3.1. Resumen de arquitecturas de computadoras paralelas.

(Pancake, Ch. M., 1996)

Características.	SIMD.	Memoria Compartida.	Memoria Distribuida.	Cluster SMP
Memoria.	Compartida.	Compartida.	Distribuida.	Compartida por los procesadores en un nodo, pero distribuida en un cluster.
Secuencia de instrucción.	Un solo flujo.	Muchos flujos.	Muchos flujos.	Muchos flujos.
Número de ejecutables.	Un ejecutable.	Un ejecutable (SPMD).	Un ejecutable o varios.	Un ejecutable o varios.
Lo más destacado.	Eficiente y relativamente fácil de programar.	Rápida y con gran memoria.	Versátil y rentable.	Versátil y rentable.
Lo menos destacado.	Se ajusta a pocos problemas.	Cara.	Difícil de usar eficientemente.	Es difícil usar muchos nodos eficientemente.
Mayor obstáculo de programación.	Usar eficientemente las operaciones con arreglos.	Proteger el acceso a datos compartidos.	Minimizar el costo de comunicación.	Proteger el acceso a datos compartidos (dentro de un nodo).

En general, cada tipo de computadora paralela es apropiada para aplicaciones con determinadas características. Si no se logra un ajuste apropiado, entre la máquina y la aplicación, se tendrá que invertir un gran esfuerzo y los resultados en el rendimiento final posiblemente sean desalentadores. Las siguientes reglas de dedo resumen la interacción entre el modelo de aplicación y el tipo de máquina **(Pancake, Ch. M., 1996)**:

1) Una aplicación de paralelización perfecta probablemente tendrá un desempeño razonablemente bueno en cualquier arquitectura MIMD, pero puede ser difícil adaptarla a una multicomputadora SIMD.
2) Una aplicación de tipo tubería probablemente tendrá su mejor rendimiento en una máquina de memoria compartida o en un cluster SMP (donde cada etapa ajusta en un solo SMP), además, puede ser adaptada a un sistema de memoria distribuida tan bien como veloz sea la red de comunicación para llevar las series de datos de una etapa a otra.
3) Una aplicación totalmente sincrónica alcanzará su mejor desempeño en una multicomputadora SIMD, si se saben manejar las operaciones con arreglos (vectores). Si las computaciones son relativamente independientes se puede obtener un rendimiento aceptable en un sistema de memoria compartida (o en un cluster

SMP si un número pequeño de procesadores es suficiente). Cualquier otro ajuste es casi imposible.

4) Una aplicación pobremente sincrónica tendrá su mejor desempeño en un sistema de memoria compartida (o un cluster SMP si un número pequeño de procesadores es suficiente). Si hay muchas computaciones entre las interacciones de los procesadores se puede tener un buen desempeño en un sistema de memoria distribuida.

3.5) Influencia del lenguaje de programación sobre el rendimiento del programa paralelo.

El lenguaje de programación también afecta al esfuerzo requerido para paralelizar una aplicación. El tipo de modelo de programación es frecuentemente el indicador clave tanto para el esfuerzo necesario como para el rendimiento final esperado.

El programador rara vez tiene opción de elegir en este aspecto. Sin embargo, excepto por las librerías, todos los lenguajes fuerzan a un modelo de programación particular. Aunque, muchas veces, también el tipo de máquina limita la decisión. Las librerías de envío de mensajes son las más ampliamente disponibles y han sido adaptadas a todas las arquitecturas MIMD.

En la **tabla 3.2** se listan los lenguajes paralelos y las librerías que están disponibles en la actualidad.

Una vez que se a determinado el tipo de aplicación y la máquina en la que se va a trabajar, la decisión del lenguaje de programación o librería a utilizar estará, seguramente, limitada a muy pocas opciones. Además, esto estará limitado por factores como: la experiencia en manejar determinado lenguaje de programación -Fortran o C-; el contacto que se tenga con personas que ya han usado el lenguaje paralelo determinado; la habilidad de usar otras

subrutinas de librería, científicas o matemáticas, que se necesiten y; la habilidad de manejar lenguajes de dominio público en el sistema, como, MPI o PVM.

Tabla 3.2. Variedades de lenguajes de programación paralela disponibles. **(Pancake, Ch. M., 1996)**

Paradigma.	Características.	Ejemplos.	Modelo.
Paralelismo de Control.	El trabajo computacional (iteraciones de ciclo o subrutinas) es dividido para ser asignado a los procesadores. Los procesadores periódicamente sincronizan sus actividades (por ejemplo, al final de un ciclo paralelo)	Parallel Fortran de IBM, VP Fortran de Fujitsu, Autotasking Fortran de Cray, ANSI X3H5 Fortran†.	MIMD / SPMD MIMD / SPMD MIMD / SPMD MIMD / SPMD
Paralelismo de Datos.	El dominio de los datos (usualmente arreglos) es subdividido y cada división es asignada a un procesador individual. Cada procesador trabaja con su serie de datos y provee copias a los otros cuando es necesario.	High Performance Fortran†, CM-Fortran de TCM, MPF de Maspar, Data Parallel C†, C* de TMC, MPL de Maspar.	MIMD / SPMD SIMD SIMD MIMD / SPMD SIMD SIMD
Envío de Mensajes.	Cada procesador trabaja por su cuenta con sus datos; cuando dos procesadores necesitan intercambiar datos o coordinar actividades, uno manda un mensaje y el otro lo debe recibir. Una librería de subrutina soporta operaciones de envío y recepción y otras más. Muy bajo nivel, virtualmente no hay detección de error.	Librería PVM†, librería MPI†, librería NX de Intel, librería p4, librería TC/MSC librería Express de Parasoft, librería MPL de IBM, Fortran M.	MIMD / MPMD MIMD / MPMD MIMD / MPMD MIMD / MPMD MIMD / MPMD MIMD / MPMD MIMD / SPMD MIMD / SPMD
Combinado.	Dos o tres de los de arriba pueden ser combinados (por ejemplo, subrutinas de paralelismo de control que se mandan mensajes entre sí).	PC++, Fortran de Convex, C de Convex.	MIMD / MPMD MIMD / SPMD MIMD / SPMD

† Estándar de lenguaje soportado, o intentado ser soportado, por muchos fabricantes de máquinas.

Los expertos dicen que existe sólo una regla de dedo en este aspecto:

1) Con pocas excepciones, "*tú no eliges al lenguaje, él te elige*".

3.6) Principales obstáculos en el progreso del paralelismo.

Los principales obstáculos que impiden el amplio uso del paralelismo caen dentro de tres categorías: hardware, algoritmos y software.

En cuanto al hardware; se está trabajando en construir redes de intercomunicación (frecuentemente llamadas switches) que puedan soportar la rapidez de los procesadores avanzados. En los últimos años se ha avanzado mucho en esto y hoy en día las máquinas tienen un mejor balance entre la computación y la comunicación.

En el área de los algoritmos; se ha avanzado en la rapidez de los programas paralelos. El avance del paralelismo de los algoritmos ha sido provisto de tres fuentes principales; de los físicos, de los matemáticos y de la imaginación de los programadores. El problema se tiene cuando se desea expresar estos nuevos algoritmos en un programa real que corra en una máquina paralela. En este punto, el problema deja de ser un problema de los algoritmos y se convierte en un problema de software.

En cuanto al software; éste es el mayor obstáculo que existe en el desarrollo de la Programación en Paralelo. El autor de un algoritmo paralelo puede encontrar, muy a menudo, que el ambiente de software obstaculiza, en vez de ayudar, al uso efectivo del hardware.

Parte de la obstrucción consiste en que, los compiladores que paralelizan automáticamente algoritmos paralelos permanecen limitados en sus capacidades. A pesar de que se ha realizado bastante investigación y los compiladores de paralelización funcionan bien con algunos programas, el mejor rendimiento es, todavía, obtenido cuando un programador por sí mismo crea el algoritmo paralelo.

Debido a este problema que, hasta la fecha, tienen los compiladores paralelos se ha buscado una solución a través de librerías de programación. Estas librerías son usadas para la programación con envío de mensajes. En este campo han habido algunos avances positivos.

Por eso, la programación con envío de mensajes ha cobrado especial importancia en los últimos tiempos.

3.7) Envío de mensajes.

Existen tres diferentes maneras de producir códigos paralelos; la paralelización semi-automática, la programación paralela de datos y la programación de envío de mensajes **(Jean-Yves, B. y Laurent, C., 1997)**.

En esta tesis se utilizó el envío de mensajes para paralelizar el simulador de Descomposición de Dominio.

El envío de mensajes es un modelo de programación paralela extensamente usado, sobre todo en máquinas de memoria distribuida. En este modelo de programación, el programa paralelo puede ser visto como una serie de instrucciones ejecutándose en paralelo sobre diferentes procesadores que intercambian mensajes durante la ejecución, para enviar o recibir ciertos valores **(Jean-Yves, B. y Laurent, C., 1997)**. Cada procesador posee una serie de datos privados y, si un procesador necesita datos de otro, le tiene que mandar un mensaje de solicitud a éste para que se los mande.

"El modelo de envío de mensajes propone una serie de procesadores que tienen sólo memoria local, pero, son capaces de comunicarse con otros procesadores enviando y recibiendo mensajes. Un factor clave en el modelo de envío de mensajes es que la transferencia de datos de la memoria local de un procesador a la memoria local de otro requiere operaciones que serán realizadas por ambos procesadores" **(Gropp, W., Lusk, E. y Skjellum, A.)**.

En el envío de mensajes las llamadas de comunicación de rutina entre procesadores son insertadas explícitamente dentro del código fuente.

La ventaja principal de las librerías de envío de mensajes es que permiten crear programas portables. La máquina virtual paralela, PVM (Parallel Virtual Machine) , y la interfase de envío de mensajes, MPI (Message Passing Interface), son los dos estándares más usados. Sin embargo, MPI se está convirtiendo en el estándar principal de las máquinas paralelas.

3.7.1) MPI.

En los 1990's se consiguieron progresos importantes en la programación de aplicaciones bajo el modelo de envío de mensajes. Debido a esto, los fabricantes de máquinas paralelas implementaron su propio ambiente de programación, lo cuál derivó en el desarrollo de software no portátil. Sin embargo, algunas librerías de programación paralela demostraron que es posible implementar un ambiente de intercambio de mensajes portátil de manera eficiente. Por lo tanto, se consideró necesario definir la sintaxis y la semántica de un conjunto de rutinas estándar que serían eficientemente implementadas a una gran variedad de computadoras. El principal objetivo del desarrollo de MPI fue la creación de dicho conjunto de rutinas estándar **(Gordillo R., J. L., 2001)**.

Las principales características y ventajas de MPI son **(Gropp, W., Lusk, E. y Skjellum, A.), (Gordillo R., J. L., 2001)**:

- No es un lenguaje, sino, una librería. Ésta especifica los nombres, secuencias de llamado y resultados de subrutinas que serán llamadas de programas en Fortran, y de funciones que serán llamadas de programas en C. Los programas que los usuarios escriben en Fortran, C y C++ son compilados con compiladores ordinarios y ligados con la librería MPI.
- Es un estándar, no una implementación en particular. Todos los fabricantes de máquinas ofrecen una implementación de MPI para sus máquinas, las implementaciones, públicas y gratuitas, disponibles deben ser capaces de correr en todas las implementaciones de MPI sin cambio alguno.
- Cumple con el modelo de envío de mensajes.

- Su estructura permite portar fácilmente códigos existentes a la Programación en Paralelo y escribir nuevos, sin tener que aprender un nueva serie de conceptos fundamentales.
- Maneja eficientemente los buffers de los mensajes. Los datos son enviados y recibidos por medio de estructuras de datos definidas por el usuario y no usando estructuras definidas internamente por las librerías de comunicación, como ocurría con ambientes paralelos anteriores.
- Permite la programación eficiente, tanto de máquinas paralelas como de grupos de estaciones de trabajo conectadas en red.
- Es totalmente portátil. El código de un programa MPI puede ser compilado y ejecutado en cualquier arquitectura sin necesidad de modificación alguna.
- Está formalmente especificado. Existe un documento oficial que contiene todas las características que debe contener una implementación estándar de MPI.

3.7.2) Conceptos básicos de MPI.

En el modelo de envío de mensajes, los procesos que se están ejecutando tienen diferentes direcciones de memoria. La comunicación se da cuando una porción de dirección de memoria de un procesador es copiada en una porción de dirección de memoria de otro. Esta operación es cooperativa y ocurre sólo cuando el primer procesador ejecuta una operación de envío y el segundo procesador ejecuta una operación de recepción. Para esto es necesario especificar algunos argumentos para las operaciones de envío y recepción. Para el procesador que envía, estos argumentos son: los datos que va a comunicar y el procesador de destino, al cual los datos serán enviados. La forma de describir los datos es especificando la dirección a la que pertenecen, el tamaño que ocupan en memoria, una asignación que indique si son enteros o reales y, el número de procesador al cual se envía. Por parte del procesador que recibe, los argumentos son: la dirección en la que se guardarán los datos, el tamaño de la memoria asignada para guardarlos, especificación del tipo de los datos –enteros o reales- y, una variable que muestre la identidad del procesador que envía –para que el procesador sepa quién le mandó la información. Otra cosa importante en el

envío y recepción de mensajes son las banderas o “tipos”, las cuales sirven para que las operaciones de envío y de recepción sean congruentes. Son como un código de seguridad para saber que lo que se recibe es realmente lo que se espera. De esta forma, una operación de recepción que tiene una cierta bandera se concluirá satisfactoriamente sólo cuando un envío de mensaje con la misma bandera llegue. Cabe señalar que, en todas las especificaciones de envío y recepción se manejan sólo números enteros.

Las principales operaciones de MPI, el envío y la recepción de mensajes, están definidas como:

send (dirección, longitud, destino, bandera)
receive (dirección, longitud, fuente, bandera)

En donde: la dirección lleva el nombre de la variable que se está comunicando (este nombre no tiene que ser forzosamente el mismo en los dos procesadores). En MPI sólo se envían variables tipo arreglo, es decir, vectores –reales o enteros-. La longitud es la dimensión de la variable. El destino y la fuente son el número del procesador que manda y del procesador que recibe, respectivamente, pues en MPI todos los procesadores tienen como nombre un número entero asignado. Y, la bandera es una identificación de los procesos de envío y recepción.

Existen muchas otras instrucciones en MPI, la cuales ayudan a mejorar la sincronización entre los procesadores, y a afinar las características de los envíos y recepciones que se desean realizar, sin embargo, las más importantes son estas dos anteriores.

CAPÍTULO IV. DESARROLLO DEL SIMULADOR.

En este capítulo se explica la forma en que se desarrolló el simulador. En una primera versión se trató la descomposición del dominio físico empleando las ideas generales expuestas en el **Capítulo II**. Como se verá mas adelante, los resultados obtenidos con este tipo de descomposición fueron satisfactorios, sin embargo, sólo se pudieron resolver problemas que involucran una descomposición en dos dominios, lo cual hizo cambiar a una descomposición sobre el sistema de ecuaciones lineales para poder abarcar un mayor número de dominios en la descomposición. En esta segunda versión fue en la que se implementó la Programación en Paralelo.

Pero, antes de mostrar el trabajo realizado en esta tesis, es conveniente conocer las características generales del simulador.

4.1) Características generales del simulador.

Se trabajó con un simulador semi-implícito de aceite negro, monofásico y en 3D **(Hernández R., O. y Del Valle M., P., 2005)**. A este simulador, que está validado y que brinda resultados confiables, se le implementó la Descomposición de Dominio y, después, se paralelizó con programación de envío de mensajes.

Las principales características del simulador son:

- La aplicación, con la que se trabajó, está programada en Fortran 90.
- Esta aplicación cumple con el modelo de paralelismo totalmente sincrónico.
- Se utilizó, la librería de envío de mensajes, MPI para paralelizar el simulador.
- La computadora paralela que se utilizó es una máquina de memoria distribuida
- El sistema operativo que tiene esta máquina paralela es Red Hat 6.0.

Teniendo en cuenta esta información, se desarrollarán los siguientes temas en este capítulo:

1. Aplicación de la Descomposición de Dominio sobre el dominio físico.
2. Aplicación de la Descomposición de Dominio sobre el sistema de ecuaciones lineales.
3. Paralelización del simulador de Descomposición de Dominio sobre el sistema de ecuaciones lineales.

4.2) Aplicación de la Descomposición de Dominio sobre el dominio físico.

Para la descomposición del dominio físico se propuso un dominio en el cual existe una zona no permeable. Inicialmente el simulador fue manipulado para generar una frontera interna de tal forma que no permitiera el flujo hacia la zona no permeable. En este caso la solución del dominio es realizada de forma tradicional, es decir, tratando el dominio como uno solo. Posteriormente se trabajó en adecuar el simulador para generar las fronteras internas que nos permitieran dividir el dominio completo, Ω_0, en dos diferentes subdominios, Ω_1 y Ω_2, y resolver estos de forma independiente.

4.2.1) Modelo propuesto.

El dominio propuesto es un dominio de longitudes 2,000 x 2,000 x 100 [pies^3], dividido en 20 x 20 x 1 celdas para las direcciones X, Y y Z, respectivamente. Las celdas localizadas en la zona [1, 11, 1] a [10, 20, 1] representan una zona impermeable, por lo que no existe flujo en esta región. El resultado de esta particularidad es un dominio con forma de "*L*", por lo que se le designo con ese nombre. Este dominio tiene dos pozos: un pozo inyector, PI, en la celda [2, 2, 1] y un pozo productor, PP, en la celda [19,19,1], los cuales tienen un gasto de 100 [bpd] cada uno. En la **figura 4.1** se muestra la vista en planta de la "*L*".

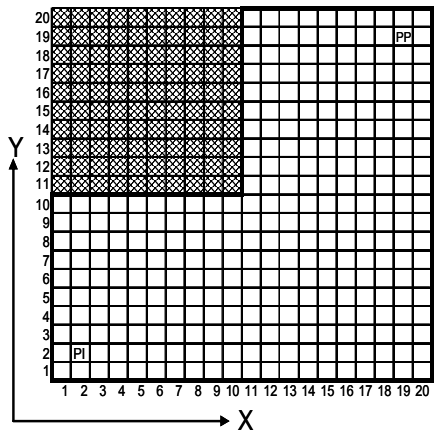


Figura 4.1. Modelo de "L" propuesto para la solución de la Descomposición de Dominio. La zona sombreada representa el área no permeable.

Para la Descomposición de Dominio se supone el siguiente esquema: dos subdominios acoplados de tal forma que tengan un traslape entre ellos. El subdominio 1 es de 1,000 x 2,000 x 100 [pies^3] dividido en 10 x 20 x 1 celdas en las direcciones X, Y y Z respectivamente, el cual tiene un pozo productor en la celda [9,19,1] con un gasto de producción de 100 [bpd]; el segundo subdominio es de 2,000 x 1,000 x 100 [pies^3] dividido en 20 x 10 x 1 celdas en las direcciones X, Y y Z respectivamente, este subdominio cuenta con un pozo inyector en la celda [2,2,1] con un gasto de inyección de 100 [bpd].

Nótese que la ubicación del pozo de cada uno de los subdominios corresponde con su ubicación en el dominio de la "*L*". El traslape para el subdominio 1, Ω_1, se encuentra en la zona de [1,1,1] a [10,10,1] y, para el subdominio 2, Ω_2, el traslape se encuentra en [11,1,1] a [20,10,1]. La **figura 4.2** muestra el esquema planteado.

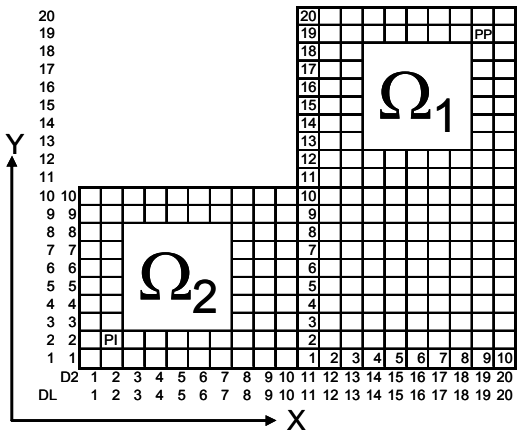


Figura 4.2. El dominio 1, Ω1, y el Dominio 2, Ω2, Componen el dominio total "L".

Las características del fluido, del yacimiento y de los pozos son resumidas en las **tablas 4.1** y **4.2**, respectivamente.

TABLA 4.1. DATOS GENERALES

L_X	2000.0
L_Y	2000.0
L_Z	100.0
N_X	20
N_Y	20
N_Z	1
N_{WP}	1
N_{WI}	1
T_{Sim}	50.0
K_X	100.0
K_Y	100.0
K_Z	100.0
μ	0.5
Gravedad API	40.0
c_t	6.9e-06
c_f	1.0e-06
ϕ	0.15
Bo	1.1
P_r	500.0
P_i	3000.0

TABLA 4.2. DATOS DE LOS POZOS

x_p	y_p	z_p	r_{wp}	Q_{op}
19	19	1	0.45	-100.0
x_i	y_i	z_i	r_{wi}	Q_{oi}
2	2	1	0.45	100.0

NOTA: La convención de signos sobre el gasto es la siguiente: Un gasto negativo representa un gasto de producción (sale del yacimiento) y un gasto positivo representa un gasto de inyección (entra al yacimiento).

4.2.2) Primera etapa del desarrollo del simulador con Descomposición del Dominio físico.

En esta primera etapa sólo se trabajó con la forma de calcular las transmisibilidades y, por lo tanto, con los coeficientes de la matriz de solución. Esto llevó a resolver un solo subdominio, considerando que no existe flujo en una región de éste.

El propósito de hacer esta "pseudo descomposición" de dominio fue el tener una herramienta de comparación que sirviera de base para decidir si los resultados son correctos, o no, cuando la solución del dominio se realiza con una descomposición de dominio real, es decir, resolviendo para presión, los dos subdominios por separado.

Dado que la transmisibilidad es el parámetro que indica la facilidad con que el flujo de fluidos se da entre dos celdas contiguas, para una celda en particular, las transmisibilidades se pueden escribir como

$$T^n_{i,j,k-\frac{1}{2}} = T_{oz1}$$
$$T^n_{i,j-\frac{1}{2},k} = T_{oy1}$$
$$T^n_{i-\frac{1}{2},j,k} = T_{ox1}$$
$$T^n_{i+\frac{1}{2},j,k} = T_{ox2}$$
$$T^n_{i,j+\frac{1}{2},k} = T_{oy2}$$
$$T^n_{i,j,k+\frac{1}{2}} = T_{oz2}$$

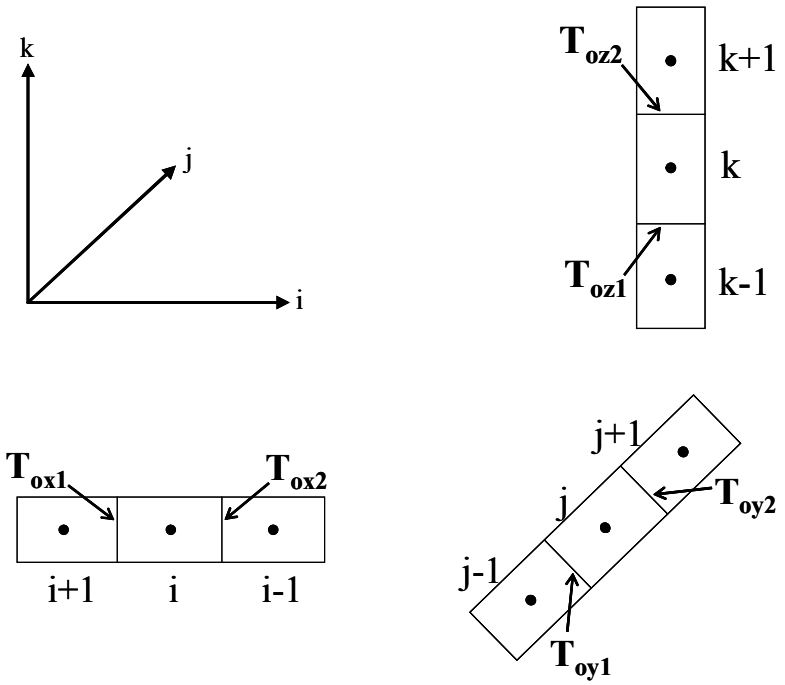


Figura 4.3. Ejemplificación de las transmisibilidades en las tres dimensiones.

Para las fronteras externas de cualquier dominio, la transmisibilidad toma el valor de cero, dado que no existe flujo en la frontera, pues, se esta considerando un yacimiento cerrado. La **figura 4.4** muestra las diferentes posibilidades para dos dimensiones. Las **figuras 4.4.a** y **4.4.b** presentan tres celdas vecinas con la celda de análisis (celda sombreada), una en X y dos en Y, en a) la celda en *i-1* no existe por lo que no hay flujo de *i-1* a *i*, por lo tanto T_{ox1} = 0, en b) la celda *i+1* no existe, por lo tanto $T_{ox2} = 0$; Las **figuras 4.4.c** y **4.4.d** presentan una situación análoga pero en la dirección Y; es decir para el subíndice en *j*.

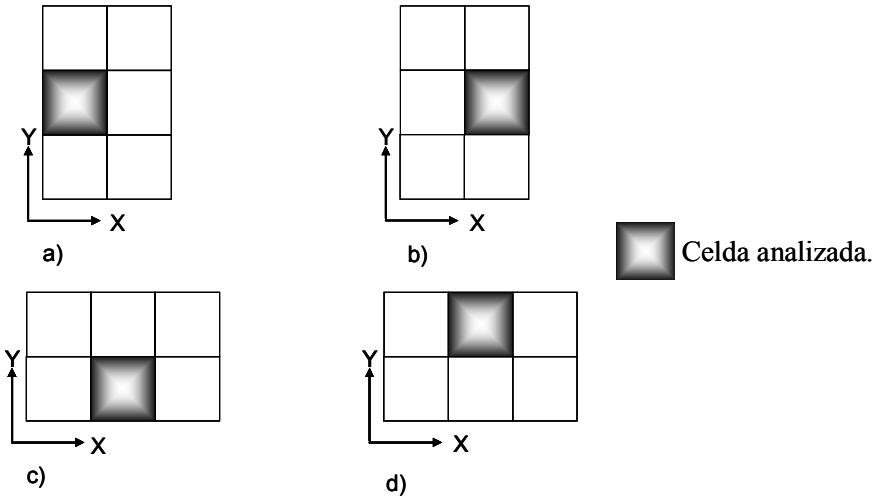
Si se considera la **figura 4.1,** para este caso, las fronteras del dominio con la zona no permeable están ubicadas en las franjas de [1,10,1] a [10,10,1] y [11,11,1] a [11,20,1]. Para estas condiciones:

$T_{ox1} = 0$ para $i = 11$, $11 \le j \le 20$

$T_{ox2} = 0$ para $i = 10$, $11 \le j \le 20$

$T_{oy1} = 0$ para $1 \le i \le 10$, $j = 11$

$T_{oy2} = 0$ para $1 \le i \le 10$, $j = 10$



a) Celda con Tox1 = 0 b) Celda con Tox2 = 0

c) Celda con Toy1 = 0 d) Celda con Toy2 = 0

Figura 4.4. Transmisibilidad en las fronteras.

Para el resto del dominio dentro de la "*L*", las transmisibilidades se calculan normalmente según la ecuación de flujo, por lo que la forma de resolver el sistema es como si se tratara de uno solo dominio.

Resultados:

El comportamiento de la presión en todo el dominio para el caso presentado se muestra en la **figura 4.5,** así mismo, para tener una referencia sobre el comportamiento de la presión a lo largo de todo el tiempo de simulación, se recurrió a la gráfica de presión de fondo fluyendo, la cual es mostrada en la **figura 4.6.** Es importante recordar que esta solución su obtuvo resolviendo un dominio, el cual fue manipulado para representar la falta de flujo en una región.

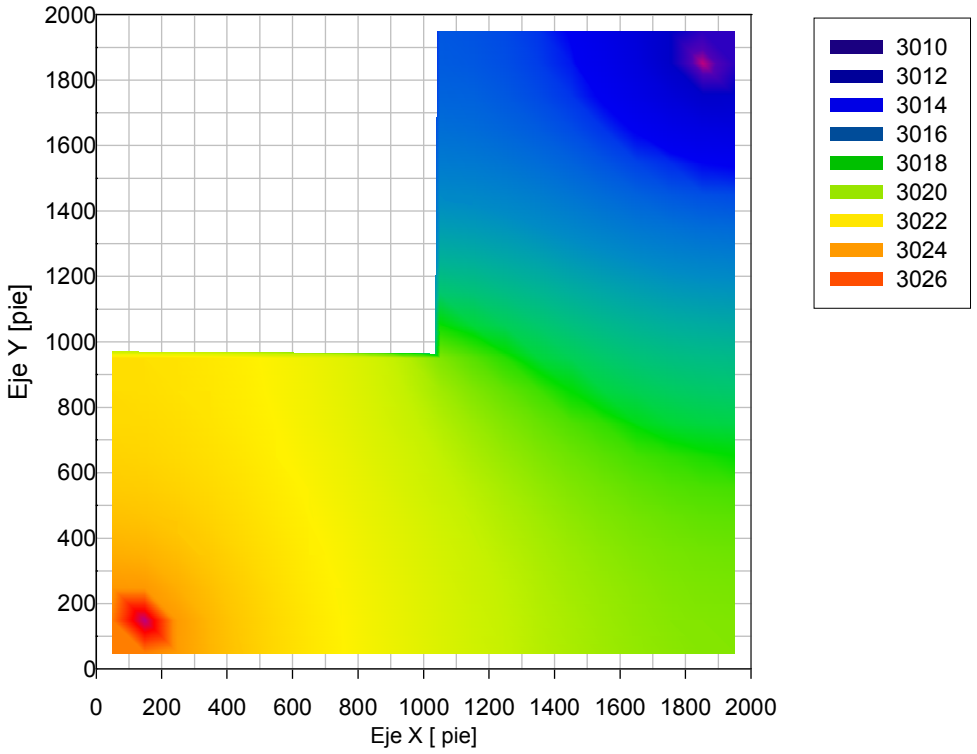


Figura 4.5. Distribución de Presión para el dominio de L. Solución de un solo dominio.

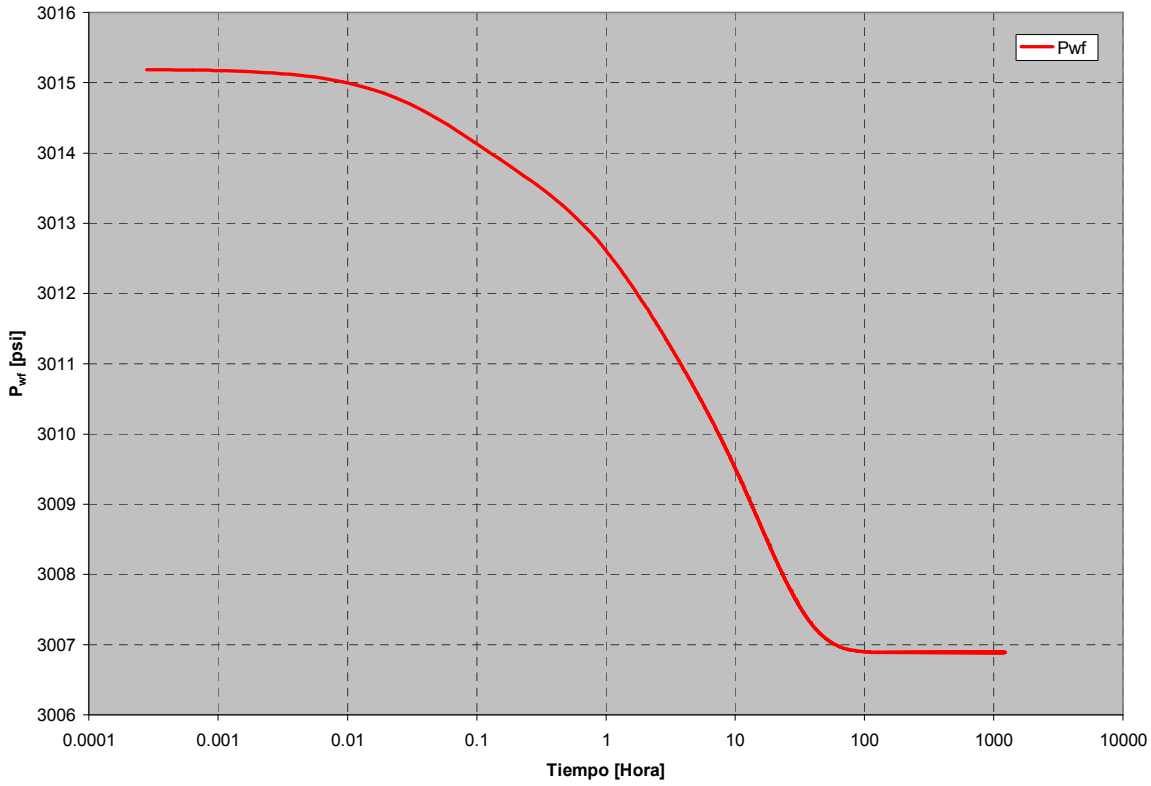


Figura 4.6. Comportamiento de Pwf para un tiempo de simulación de 50 días.

4.2.3) Segunda etapa del desarrollo del simulador con Descomposición del Dominio físico.

En esta sección se muestra la forma en que se trató el problema para resolver el dominio completo a través de dos subdominios. Es decir, encontrar la solución global a partir de la solución independiente de cada subdominio.

Al establecer la ecuación de flujo para cada una de las celdas en la malla de simulación y resolver el sistema de ecuaciones para un paso de tiempo, Δt, se lleva la presión actual, P^n, a un paso de tiempo adelante, por lo que la presión nueva obtenida es P^{n+1}. Si resolvemos el sistema de ecuaciones para cada uno de los subdominios se obtendrá una P^{n+1} en cada una celdas para cada uno de los subdominios. Dado que los dos subdominios son resueltos independientemente, la presión en la zona del traslape es diferente para cada uno de ellos. Para igualar la presión en la zona del traslape entre los dos subdominios se recurre a establecer las condiciones de frontera sobre la interfase para los dos subdominios considerando lo siguiente.

A cada una de las celdas *i, j, k* en la interfase del dominio 1, le corresponde una celda *i', j', k'* del subdominio 2, y viceversa, la cual representa la misma celda sobre el dominio total, ver **figura 4.2.** Si se observa esta situación en la **figura 4.7.,** se notará que entre la celda *i, j, k* del subdominio 1 y la celda *i'*-1, *j', k'* del subdominio 2 debería de existir una transmisibilidad puesto que en el dominio total existe comunicación entre estas celdas. Sin embargo, dado que en la interfase se esta considerando que existe una frontera impermeable, en lugar de determinar una transmisibilidad, lo que se determina es el gasto que debería de pasar de una celda a otra y este gasto es empleado como condición de frontera. Con la diferencia de presión entre las dos celdas en cuestión se puede determinar un pseudogasto, Q'_o, que debería pasar de una celda a otra, o lo que es lo mismo, si se consideran todas las celdas de la interfase, el gasto que debería pasar de un subdominio a otro.

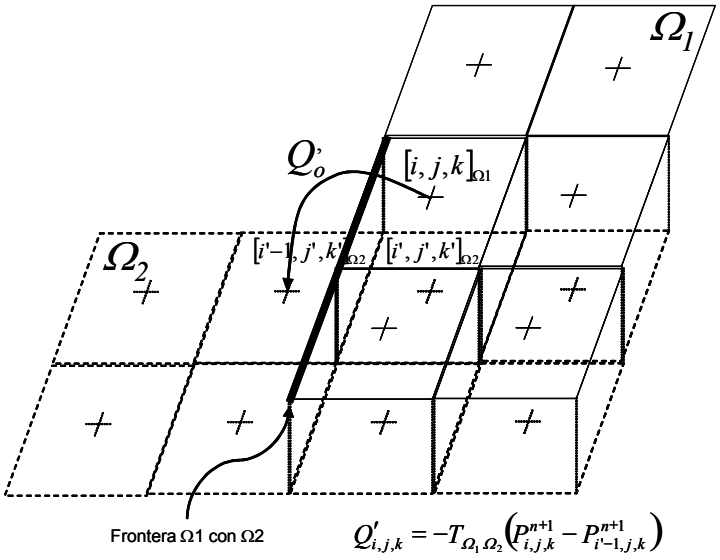


Figura 4.7. El gasto en la dirección de x que debería de existir en la frontera de Ω1 es calculado con la diferencia de presión en las celdas (i,j,k) y (i'-1,j',k').

El pseudo gasto, Q'_o, es calculado para cada una de las celdas en la interfase a partir de la **ecuación (4.1)**.

$$Q'_o = T_{\Omega 1\Omega 2}\left(P^{n+1}_{\Omega 2} - P^{n+1}_{\Omega 1}\right)_{i,j,k} \qquad \textbf{…(4.1)}$$

Donde:

$$T_{\Omega 1\Omega 2} = \frac{\Delta y \Delta z}{\Delta x}\frac{k}{\mu B}$$

es evaluada corriente arriba.

Este procedimiento es repetido para el subdominio 2. Se debe notar que cuando se calcula Q'_o para el subdominio 1, la transmisibilidad a la que se hace referencia es sobre la dirección *x*, y cuando el Q'_o es calculado para el subdominio 2, la transmisibilidad es referida a la dirección y.

Una vez que el pseudo-gasto en la interfase es establecido se determina la matriz de coeficientes y el vector de términos independientes para cada subdominio y los sistemas son nuevamente resueltos de forma independiente.

El procedimiento anterior es iterativo hasta que la máxima diferencia de presión obtenida de cada una de las celdas del traslape cae dentro de una cierta tolerancia, definida por la **expresión (4.2)**.

$$\text{Tol} \geq \left[P_{\Omega 2}^{n+1} - P_{\Omega 1}^{n+1} \right]_{Max} \qquad \textbf{…(4.2)}$$

Resultados:

Básicamente los datos empleados son los mismos que los mostrados en las **tablas 4.1** y **4.2**, la única diferencia es que en esta ocasión dado que ahora se están tratando dos subdominios, en vez de uno global, se requiere información sobre éstos. Esta información es presentada en las **tablas 4.3** y **4.4**.

TABLA 4.3. DATOS DE LOS SUBDOMINIOS

Dominio	N_x	N_y	N_z	C_x	C_y	C_z	N_{WP}	N_{WI}
1	10	20	1	11	1	1	1	0
2	20	10	1	1	1	1	0	1

TABLA 4.4. DATOS DE LOS POZOS EN LOS SUBDOMINIOS

Dominio	X_p	Y_p	Z_p	r_{wp}	Q_{op}
1	9	19	1	0.45	-100.0
2	-	-	-	-	-
	X_i	Y_i	Z_i	r_{wi}	Q_{oi}
1	-	-	-	-	-
2	2	2	1	0.45	100.0

Las distribuciones de presión obtenidas a 50 días para cada uno de los dos subdominios y para el dominio completo, son mostradas en la **figura 4.8**.

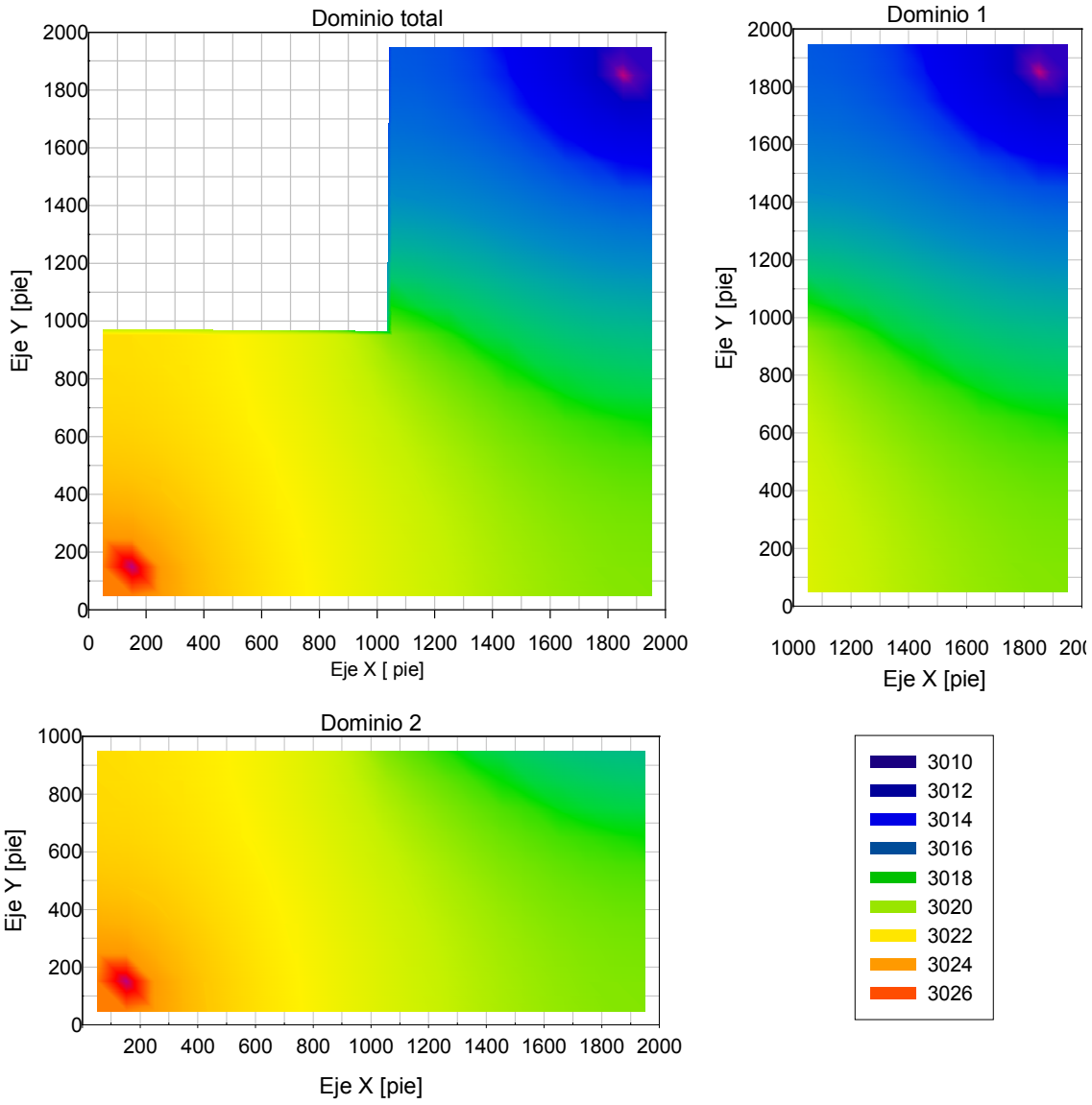


Figura 4.8. Distribución de presión a 50 días para dos subdominios y su correspondiente dominio total. Solución obtenida a partir de un tratamiento separado en los dos subdominios.

Como se puede apreciar en la **figura 4.8**, los resultados obtenidos al descomponer en dos subdominios son idénticos a los obtenidos resolviendo el sistema como uno solo. Además, como se puede ver en la gráfica de P_{wf} contra el tiempo, que se presenta en la **figura 4.9,** el comportamiento de las dos soluciones también es idéntico durante todo el tiempo de simulación.

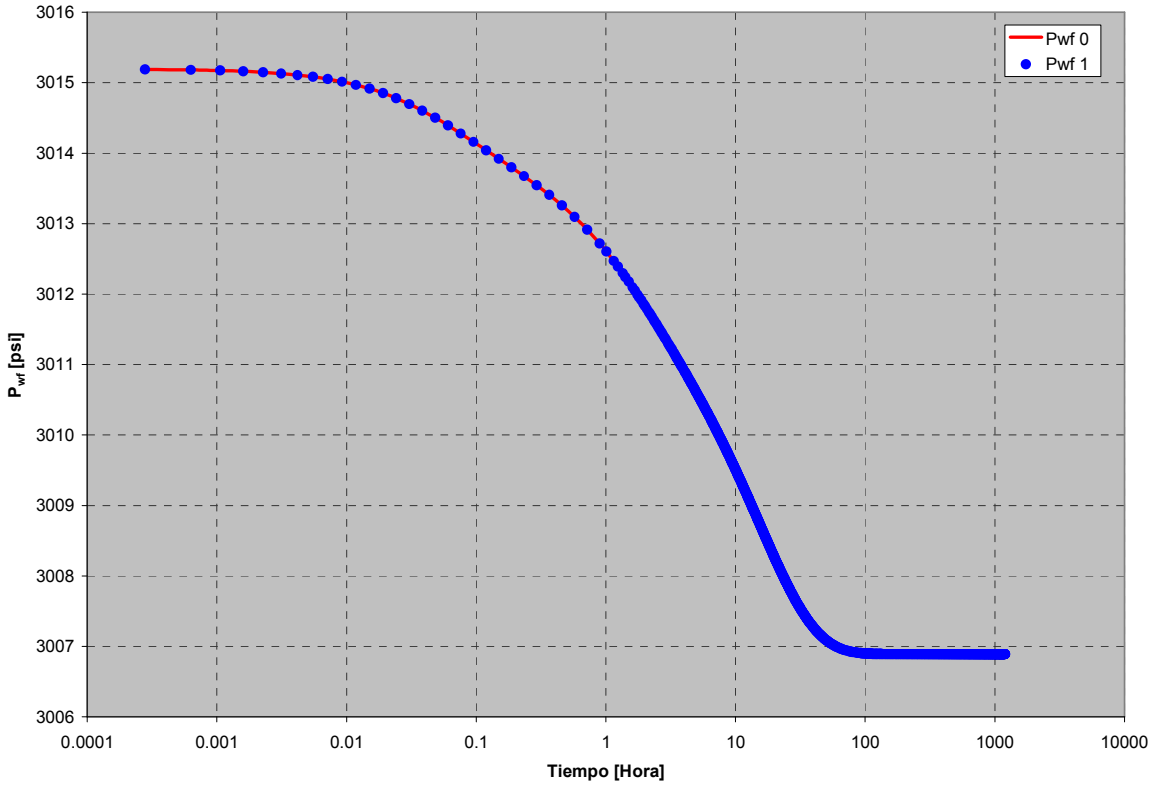


Figura 4.9. Comparativo entre Pwf del dominio completo (Pwf 0) y la Pwf obtenida del subdominio 1 (Pwf 1).

Aparte de este caso, se probó el simulador con otros esquemas. A continuación se presenta uno más, representado por la **figura 4.10**. Los datos empleados para este segundo caso son mostrados en las **tablas 4.5 – 4.7**.

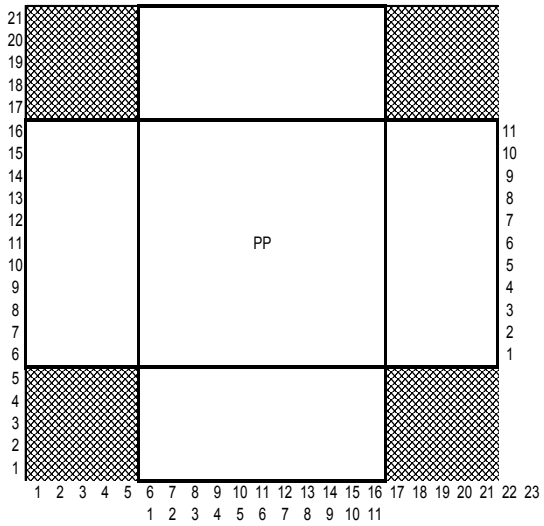


Figura 4.10. Modelo para la solución de la Descomposición de Dominio. Las zonas sombreadas representas áreas no permeables.

TABLA 4.5. DATOS GENERALES

L_X	2100.0
L_Y	2100.0
L_Z	100.0
N_X	21
N_Y	21
N_Z	1
N_{WP}	1
N_{WI}	0
T_{Sim}	50.0
K_X	100.0
K_Y	100.0
K_Z	100.0
μ	0.5
Gravedad API	40.0
c_t	6.9e-06
c_f	1.0e-06
ϕ	0.15
Bo	1.1
P_r	500.0
P_i	3000.0

TABLA 4.6. DATOS DE LOS POZOS

Dominio	X_p	Y_p	Z_p	r_{wp}	Q_{op}
0	11	11	1	0.45	-100.0
1	11	6	1	0.45	-100.0
2	6	11	1	0.45	-100.0

TABLA 4.7. DATOS DE LOS SUBDOMINIOS

Dominio	N_x	N_y	N_z	C_x	C_y	C_z	N_{WP}	N_{WI}
1	21	11	1	1	6	1	1	0
2	11	21	1	6	1	1	1	0

Los resultados obtenidos para este caso, en relación a la distribución de la presión en el yacimiento y al comportamiento de la presión de fondo fluyendo contra el tiempo, son mostrados en las **figuras 4.11 y 4.12**, respectivamente.

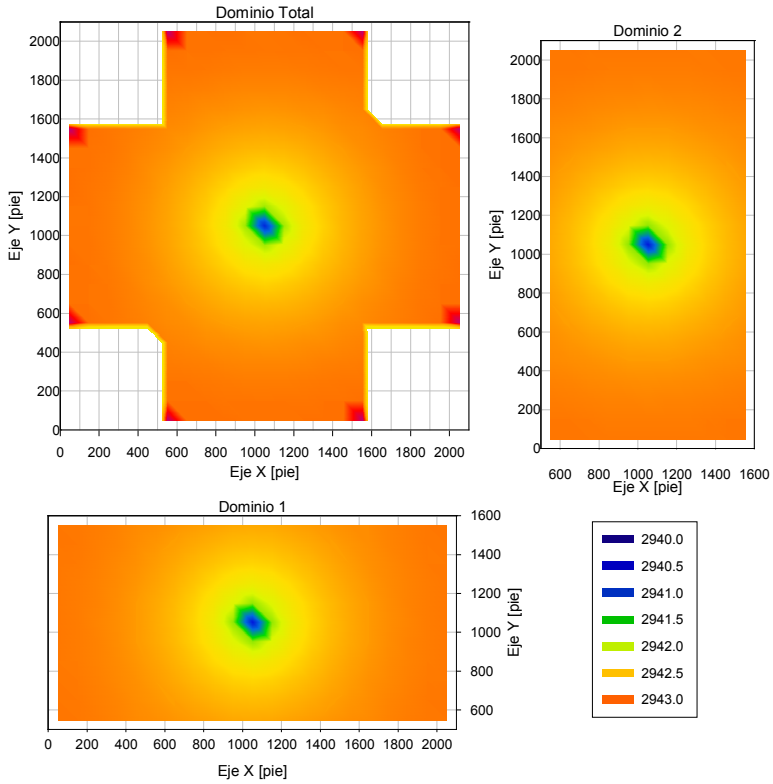


Figura 4.11. Distribución de presión a 50 días para dos subdominios y su correspondiente dominio total. Solución obtenida a partir de un tratamiento separado en los dos subdominios.

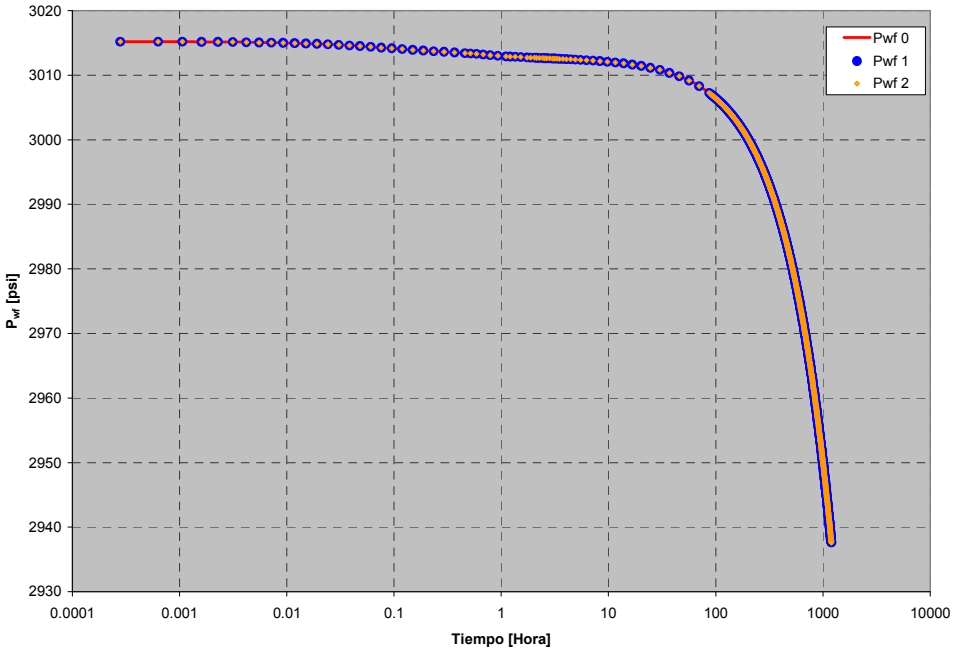


Figura 4.12. Comparativo entre Pwf del dominio completo (Pwf 0) y la Pwf obtenida para los dos subdominios (Pwf 1, Pwf 2).

Como se puede observar en las gráficas presentadas para este segundo ejemplo, de igual forma que en el primer caso, se obtuvieron excelentes ajustes en la distribución de presión y en el comportamiento de la presión de fondo fluyendo. Las diferencias presentadas en las esquinas (en rojo) del dominio completo se deben a la diferencia en la interpolación del software empleado para la visualización gráfica.

En un tercer caso, se intento hacer la simulación de un yacimiento en forma de "*S*" descompuesto en tres subdominios, como el mostrado en la **figura 4.13.**

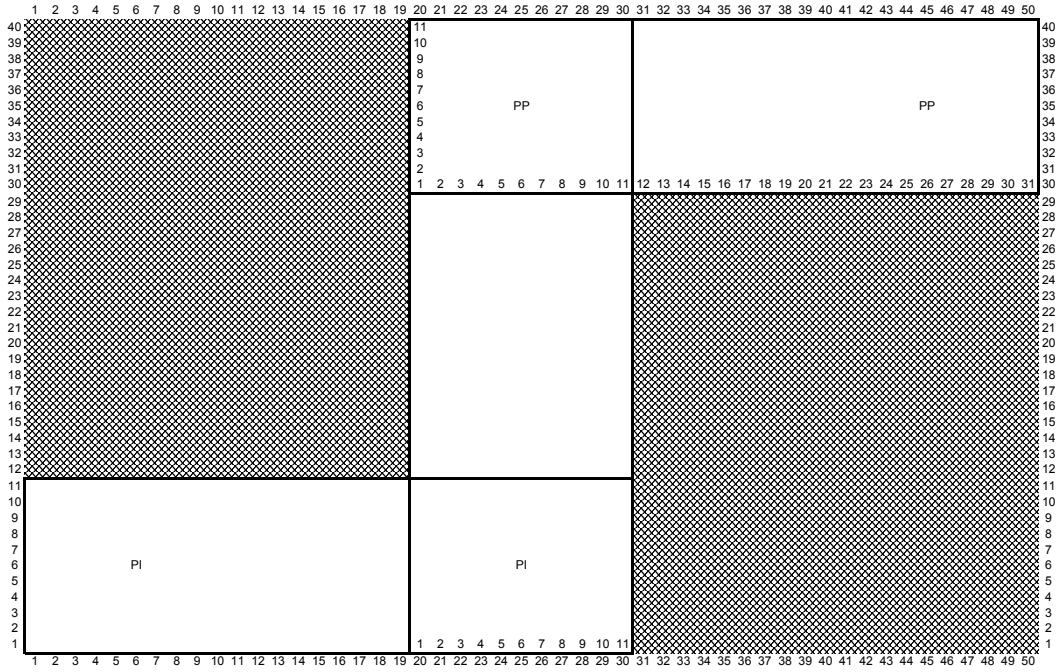


Figura 4.13. Modelo para la solución de la Descomposición de Dominio. Las zonas sombreadas representas áreas no permeables.

Los resultados para este tercer caso no son mostrados, debido a que, no se logró la convergencia adecuada sobre las zonas de traslape.

La dificultad para poder dividir un yacimiento en más de dos subdominios hizo cambiar el trabajo a otro tipo de Descomposición de Dominio, ahora sobre el sistema de ecuaciones lineales. Este tipo de descomposición es tratado en la siguiente sección.

4.3) Aplicación de la Descomposición de Dominio sobre Sistema de Ecuaciones Lineales.

En este tipo de descomposición se va a abundar más que con la Descomposición sobre el dominio físico, pues fue con la que más se trabajó en el desarrollo de esta tesis y, además, fue con ésta con la que se paralelizó el simulador. Debido a la cantidad predominante de figuras y tablas en comparación al texto que las describe y explica, se ha decidido mostrar las figuras y tablas al final de la explicación correspondiente a cada análisis.

Para aplicar la Descomposición de Dominio sobre el sistema de ecuaciones lineales se creó una subrutina en Fortran 90, llamada Schwarz, encargada de realizar el método multiplicativo de Schwarz. En esta subrutina se divide el sistema de ecuaciones global en subsistemas y se resuelve iterativamente para cada paso de tiempo.

Se corrió el programa y se analizaron sus resultados contra los obtenidos con el simulador sin Descomposición de Dominio –simulador original- para evaluar la confiabilidad de los resultados. A continuación se muestran los análisis y mejoras, más importantes, que se hicieron para afinar el desempeño del simulador y, así, obtener resultados más exactos y en menor tiempo.

4.3.1) Análisis sobre el ancho de banda.

Cuando la ecuación de flujo discretizada es aplicada sobre cada uno de los bloques que forman la malla numérica, se forma un sistema de ecuaciones lineales, de la forma $Ax = b$, que debe ser resuelto simultáneamente. En el caso de un modelo en tres dimensiones con ordenamiento natural de las celdas el sistema resultante es un sistema disperso formado por siete bandas. En otras palabras, la mayoría de los elementos de la matriz de coeficientes son cero, con excepción de ciertas bandas (siete, en este caso). En la **figura 4.14.a**, se muestra el modelo en 3D y el esquema de la matriz de coeficientes, para un caso con $Nx = 4$, $Ny = 3$, $Nz = 2$.

El ordenamiento natural se realiza recorriendo la malla primero sobre *x* (1 a *Nx*), luego sobre *y* (1 a *Ny*), y, por último, se hace el recorrido sobre *z* (1 a *Nz*). El ancho de banda esta definido como la máxima amplitud de cualquier renglón entre las dos bandas más externas; para el modelo en tres dimensiones propuesto se puede obtener como $2(Nx)(Ny)+1$.

Si el ordenamiento de las celdas de la malla numérica se realiza de diferentes formas, el ancho de banda también es diferente. Para estos casos, la forma de obtener el ancho de banda es $2(Na)(Nb)+1$, donde *Na* y *Nb*, son el primer y segundo índices, respectivamente, en que se recorre la malla. En las **figuras 4.14.a** – **4.14.f** se muestran las diferentes formas de ordenar la malla y el esquema resultante de la matriz de coeficientes.

Como se puede observar, en la **figura 4.14.f** se tiene el menor ancho de banda. Esto no quiere decir que siempre se tendrá el menor ancho de banda con un ordenamiento (*k, j, i*). Si se es cuidadoso, se notará que el menor ancho de banda se logra cuando el ordenamiento se realiza del menor al mayor número de bloques (para cada una de las tres direcciones) de la malla numérica. Esto es importante por que será retomado más adelante.

En el presente análisis, se evaluó el impacto que tiene el ancho de banda sobre la solución del sistema, aplicando el método multiplicativo de Shwarz y utilizando NSPIV como método para resolver los sistemas de ecuaciones lineales. Para tal propósito, se consideraron diferentes modelos en tres dimensiones de forma que la dimensión del sistema de ecuaciones lineales global resultante fuera constante e igual a 1620x1620 (esto, debido a que el programa original no tiene la capacidad de ordenar los índices N_x, N_y y N_z de acuerdo a su tamaño). Los datos generales para la simulación se muestran en la **tabla 4.8**. Cabe hacer notar que las longitudes en las direcciones *x, y* y *z,* son variables en cada caso. Esto se hizo con la finalidad de tener celdas de tamaño constante –100 [pie] por lado-. Los casos empleados en el análisis son mostrados en la **tabla 4.9**.

En todos los casos el sistema fue evaluado con el método multiplicativo de Shwarz, aplicando una Descomposición de Dominio en cuatro subsistemas. El número de

iteraciones máximas para Shwarz fue de 20 con una tolerancia de 1.0E-03 y un traslape de 20 renglones hacia cada extremo del subsistema.

El parámetro "número de iteraciones máximas" define el número máximo de iteraciones que tiene la subrutina Schwarz para llegar a la tolerancia deseada en la solución del sistema global. Si se llega a este número de iteraciones y todavía no se alcanza la convergencia, el programa se sale de la subrutina, divide el incremento de tiempo (Δt), actualiza los coeficientes y vuelve a entrar a la subrutina Schwarz para calcular otra vez y llegar a la convergencia. Esto se hace durante un paso de tiempo y bajo el criterio de que 20 iteraciones son suficientes para alcanzar la convergencia y, si no se alcanza en éstas lo mejor es disminuir el Δt y regresar a los cálculos. El parámetro "traslape" indica la cantidad de columnas y renglones que cada subsistema compartirá con los demás, como mínimo.

Los parámetros observados en este análisis fueron:

- El Ancho de banda del sistema.

- La distancia (en número de elementos) que existe de la diagonal principal hacia cada una de las diagonales superiores (la distancia hacia las diagonales inferiores es recíproca).

- Numero total de pasos de tiempo empleados en la simulación.

- Tiempo total de corrida, en segundos, denotado como *Run Time* (este no considera el tiempo de lectura ni el tiempo de asignación de condiciones iniciales).

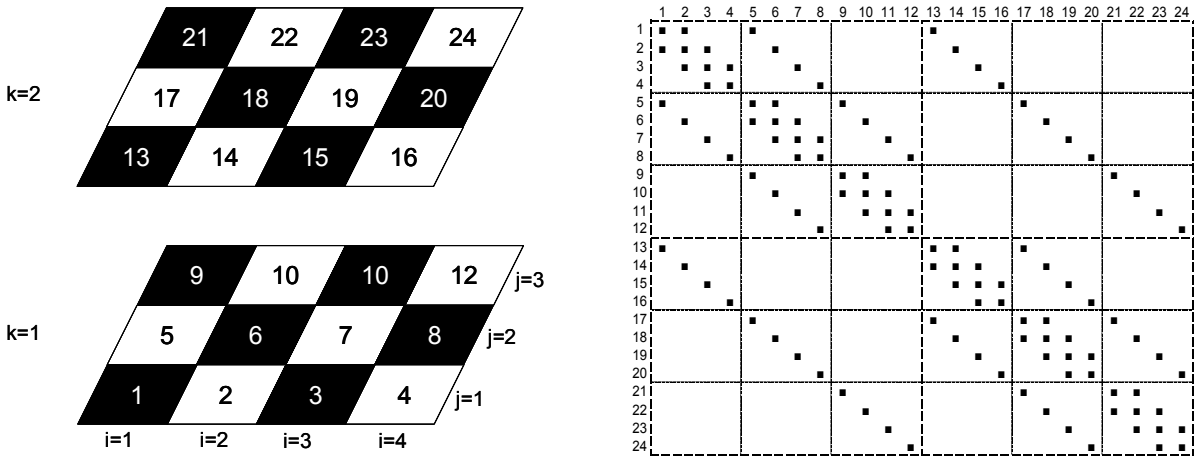
- El tiempo promedio, en segundos, empleado en la solución de cada uno de los subsistemas.

En la **tabla 4.10** se muestran los resultados tabulados obtenidos para los seis casos evaluados y en las **figuras 4.15.a – 4.15.c** se muestran las gráficas correspondientes al número de pasos de tiempo (NTS), al tiempo de corrida (Run Time) y al Tiempo promedio de solución de cada subsistema (In Time). En las **figuras 4.16.a – 4.16.c** se muestran las soluciones obtenidas del simulador con Descomposición de Dominio (Shwarz-NSPIV) comparándose con las obtenidas con el simulador original (NSPIV).

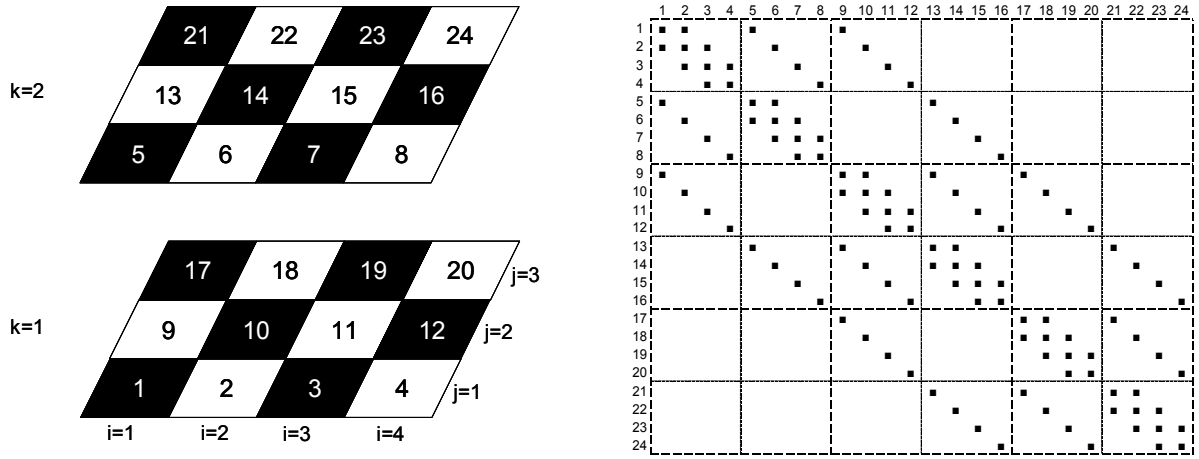
De las **figuras 4.15.a - 4.15.c**, se puede observar que los mejores valores, para los tres parámetros observados, se obtienen para los casos 4 y 6, es decir con arreglos de 12x9x15 y de 9x12x15. Si se observa el ancho de banda, *bw,* obtenido en estos dos casos y se compara con el ancho de banda de los otros casos, se puede notar que para estos dos se tiene el menor ancho de banda posible. Además, sobre estos dos, el que ofrece los menores valores es el caso 6. En base a estas observaciones se puede concluir que cuando el ancho de banda es el menor posible, los tiempos computacionales requeridos se reducen. Esto es logrado cuando el ordenamiento de la malla es del menor al mayor número de bloques en la malla de simulación. Es decir, cuando se comienza a leer la malla por el lado que tiene menos bloques y se termina de leer por el que tiene más bloques.

En las **figuras 4.16.a - 4.16.c** se observa que el ancho de banda no afecta la exactitud de la solución, por lo tanto, el esfuerzo para tener el menor ancho de banda está orientado sólo a optimizar el tiempo de corrida y no los resultados.

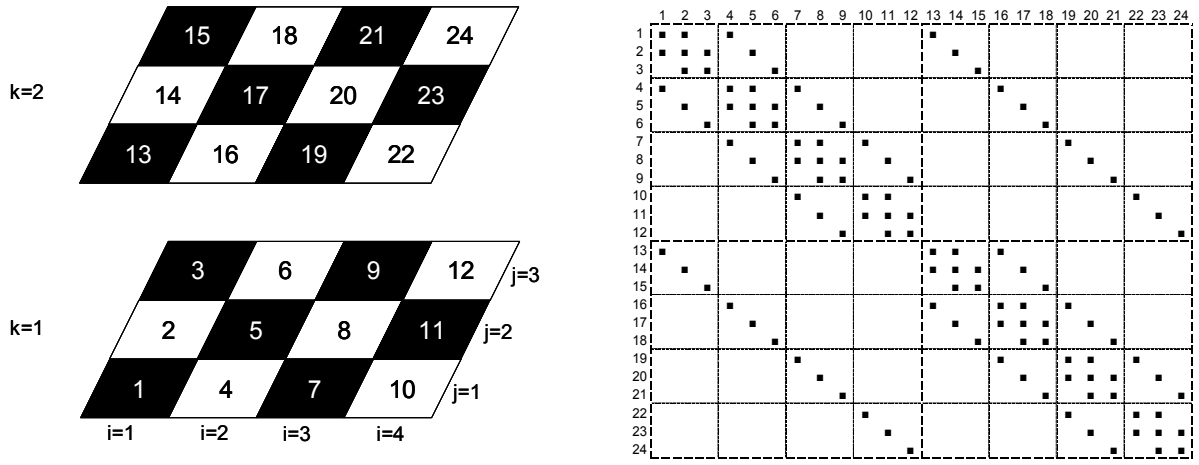
NOTA: la confiabilidad de los resultados se pudo haber evaluado en 3D, analizando el comportamiento de la presión en todo el yacimiento, como en el caso de la Descomposición sobre el dominio físico. Sin embargo, se prefirió utilizar la gráfica de P_{wf} contra tiempo, debido a que, se utilizó el criterio de que si la gráfica de P_{wf} contra tiempo (obtenida con el simulador con Descomposición de Dominio) es igual o muy cercana a la gráfica de P_{wf} contra tiempo base (obtenida con el simulador original), entonces la distribución de presiones en el yacimiento también sería igual o muy cercana a la distribución de presión base. Además, en los casos en que se utilizan pozos productores e inyectores en la simulación, se utiliza sólo la P_{wf} de los pozos productores.



a) Ordenamiento natural. (i, j, k)

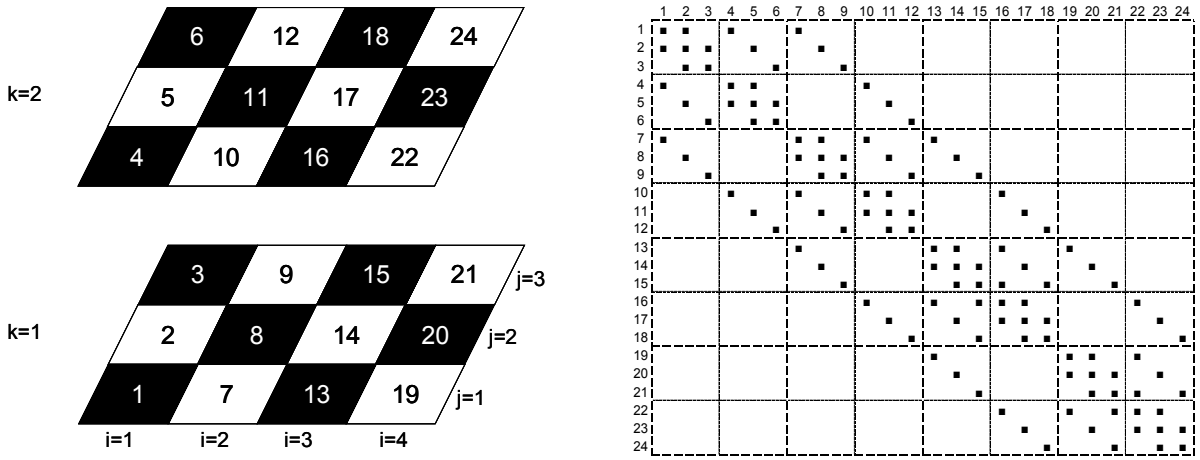


b) Ordenamiento (i, k, j)

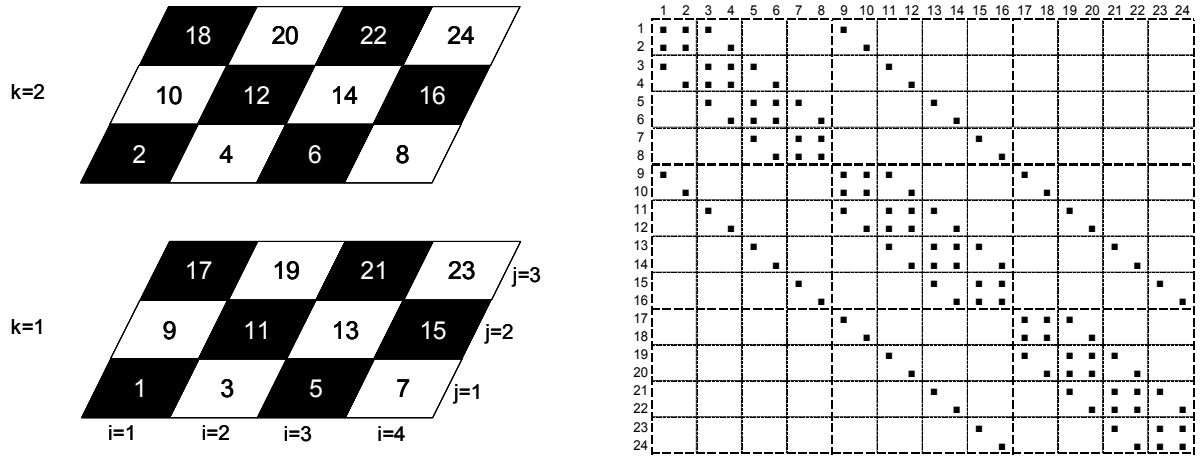


c) Ordenamiento (j, i, k)

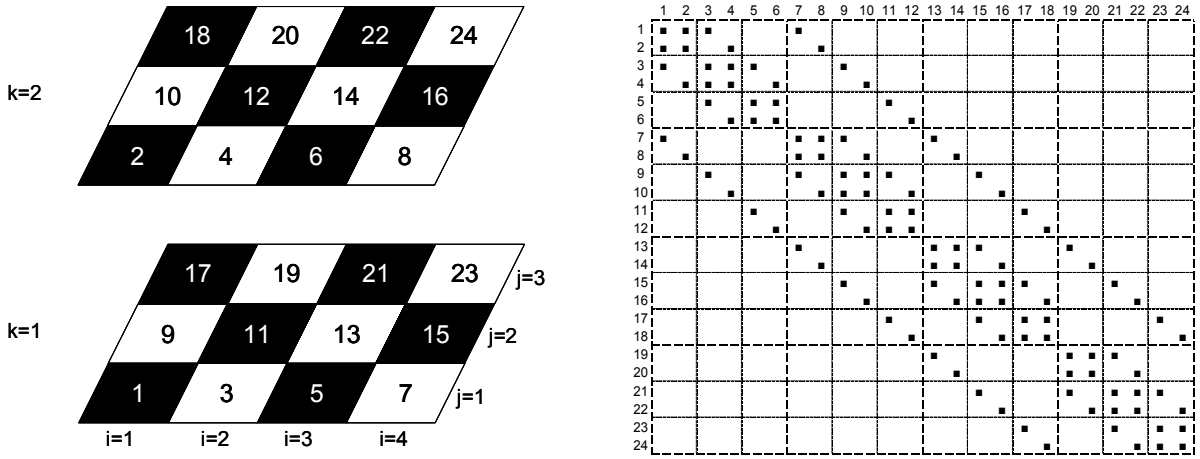
Figura 4.14. Modelo en 3D con Nx = 4, Ny = 3, Nz = 2. y su correspondiente esquema de la matriz de coeficientes resultante.



d) Ordenamiento (j, k, i)

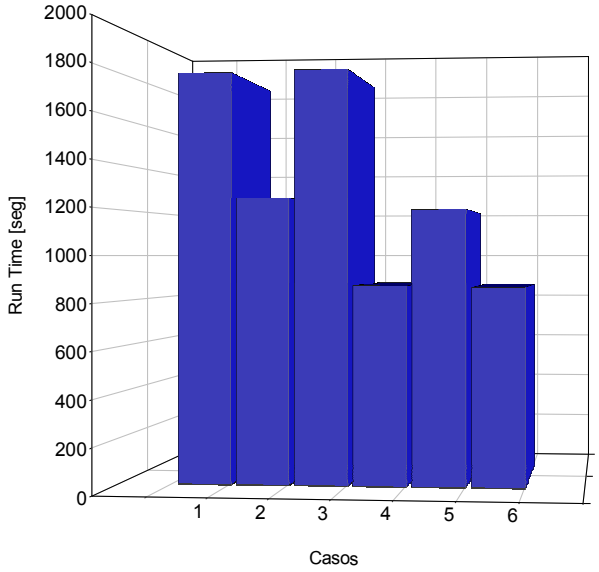


e) Ordenamiento (k, i, j)

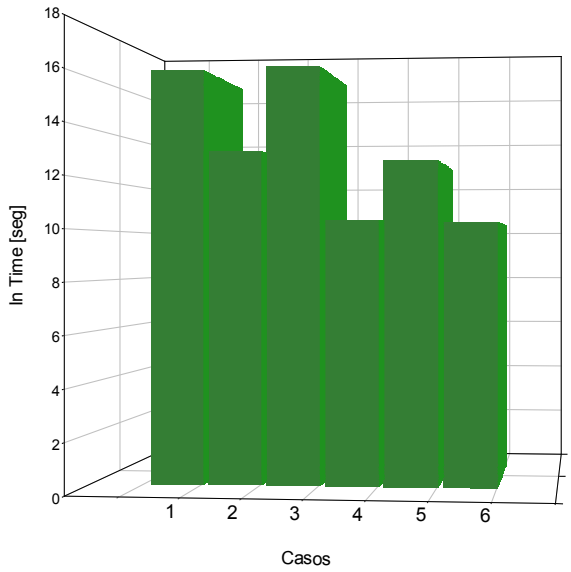


f) Ordenamiento (k, j, i)

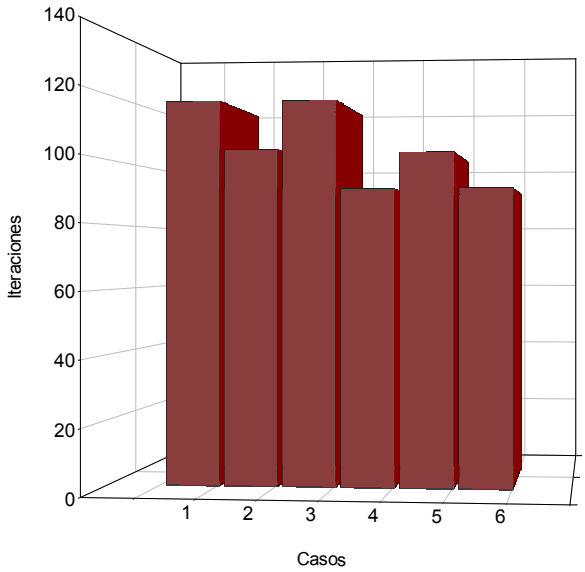
Figura 4.14. Modelo en 3D con Nx = 4, Ny = 3, Nz = 2. y su correspondiente esquema de la matriz de coeficientes resultante.



a) Tiempo de corrida.

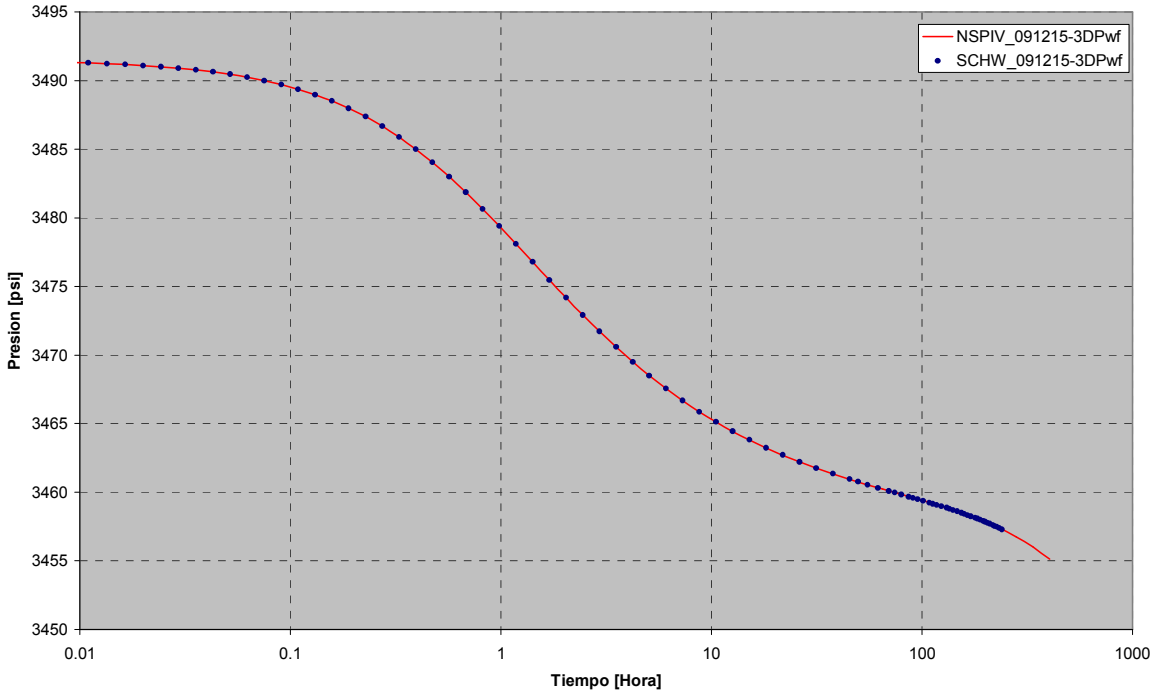


b) Tiempo de solución de cada subsistema.

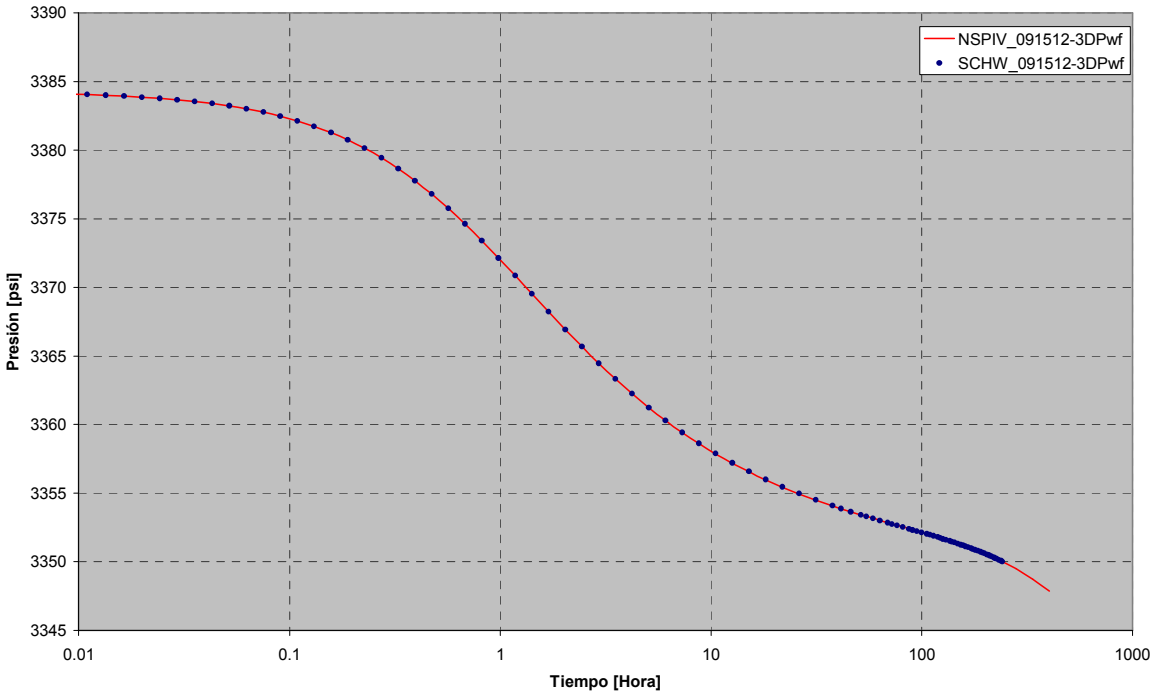


c) Numero de pasos de tiempo.

Figura 4.15. Valores obtenidos para los 6 casos evaluados con Shwarz-NSPIV.

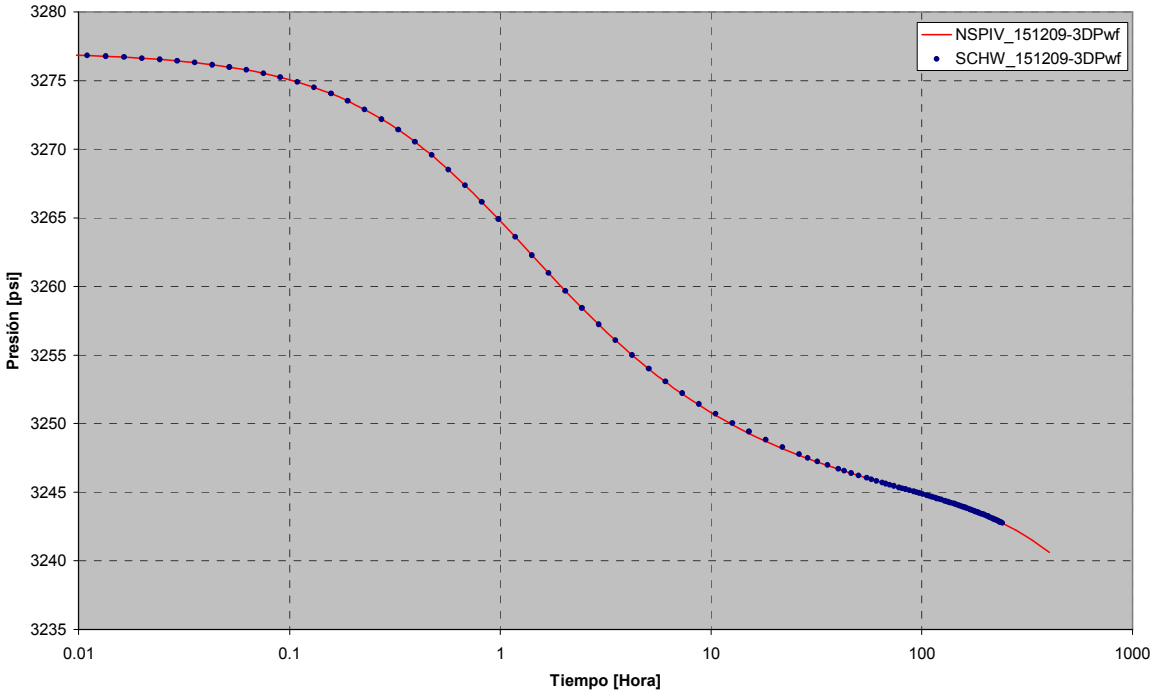


a) Pwf con ancho de banda de 217.



b) Pwf con ancho de banda de 271

Figura 4.16. Soluciones obtenidas para un sistema de ecuaciones de 1620x1620 con ancho de banda variable.



c) Pwf con ancho de banda 361.

Figura 4.16. Soluciones obtenidas para un sistema de ecuaciones de 1620x1620 con ancho de banda variable.

Tabla 4.8. Datos del yacimiento, fluido y pozo productor empleados en la simulación numérica para el caso del análisis del ancho de banda.

Datos Generales del problema			
L_X	: 100.0*N_x^*	μ	: 0.5
L_Y	: 100.0*N_y^*	API	: 40.0
L_Z	: 100.0*N_z^*	C_f	: 6.9E-06
N_x	: N_x^*	C_r	: 1.0E-06
N_y	: N_y^*	ϕ	: 0.15
N_z	: N_z^*	Bo	: 1.1
N_{WP}	: 1	P_r	: 500.0
N_{WI}	: 0	P_i	: 3000.0
T_{sim}	: 15.0		
K_X	: 10.0		
K_Y	: 10.0		
K_Z	: 10.0		
Datos del pozo productor			
X_P	: 1		
Y_P	: 1		
Z_P	: 1		
r_{wp}	: 0.45		
Q_o	: 100.0		

* Los valores de *NX, Ny* y *Nz*, son tomados para cada uno de los casos de la tabla 4.9.

Tabla 4.9. Casos Analizados para el ancho de banda.

Casos	N_x	N_y	N_z	N_c
1	15	12	9	1620
2	15	9	12	1620
3	12	15	9	1620
4	12	9	15	1620
5	9	15	12	1620
6	9	12	15	1620

Tabla 4.10. Resultados para Shwarz-NSPIV con variación del ancho de banda.

#	N_x	N_y	N_z	N_c	bw	GapE	GapN	GapU	NTS	Run Time	In Time
1	15	12	9	1620	361	1	15	180	118	1800.56	16.27
2	15	9	12	1620	271	1	15	135	103	1252.02	13.14
3	12	15	9	1620	361	1	12	180	118	1813.13	16.40
4	12	9	15	1620	217	1	12	108	91	867.50	10.38
5	9	15	12	1620	271	1	9	135	102	1200.09	12.74
6	9	12	15	1620	217	1	9	108	91	862.06	10.32

4.3.2) Descomposición de Dominio vs. Solución Directa.

De acuerdo a la teoría, con la Descomposición de Dominio se incrementa la capacidad de cómputo de las máquinas. En este análisis se muestra la validez de esta afirmación.

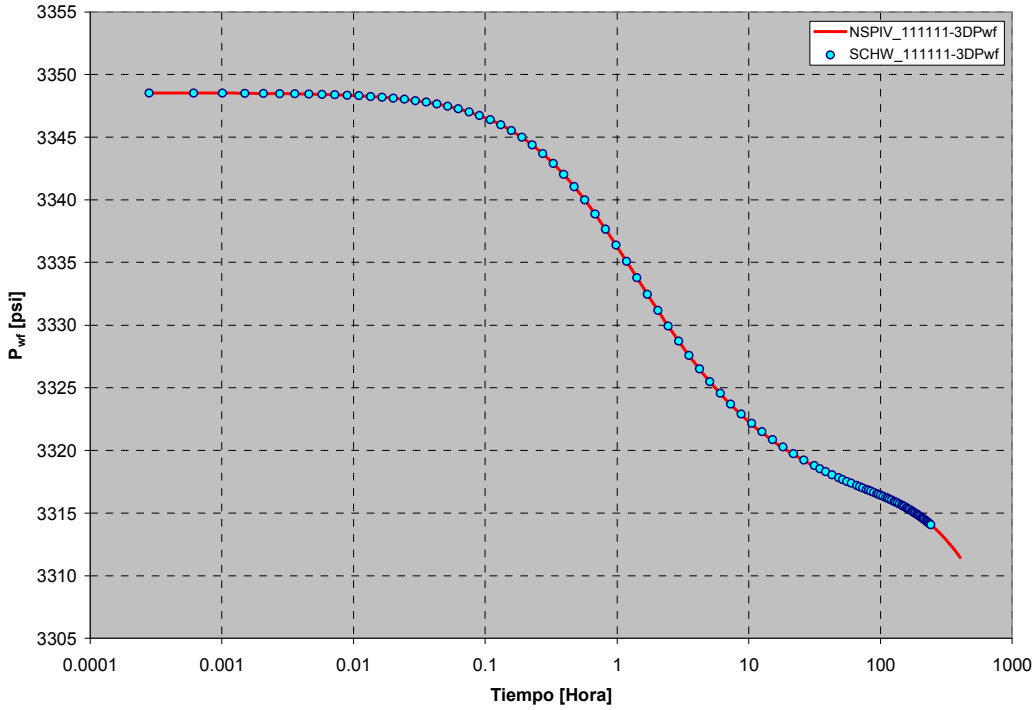
El simulador original utiliza el método NSPIV para resolver el sistema de ecuaciones lineales. En el simulador con Descomposición de Dominios también se utiliza NSPIV para resolver cada uno de los subsistemas generados. En ambos simuladores, NSPIV es una subrutina que emplea eliminación gaussiana dispersa en la solución de sistemas dispersos del tipo $Ax = b$.

Para poder mostrar el potencial de la Descomposición de Dominio, Shwarz, se decidió limitar el número máximo de elementos empleados en la solución con NSPIV. Es decir, se limitó la dimensión máxima del sistema que puede resolver la subrutina NSPIV, simulando así un límite máximo (imaginario) para el cual NSPIV puede resolver sistemas o, bien, el límite máximo de cómputo de la máquina. Para realizar esto, se restringió la subrutina de forma que resolviera sistemas de dimensión máxima de 1000 x 1000. Los sistemas a resolver fueron obtenidos a partir de un modelo en 3D de forma cúbica, de tal forma que el ancho de banda no interfiriera con la solución, por lo tanto, $N_x = N_y = N_z$ y, entonces, el tamaño del sistema, N_c, es igual a $(N_x)^3$. En las corridas con el simulador de Descomposición de Dominio se utilizó un traslape de 40, 20 iteraciones máximas para llegar a la convergencia y, el número de subdominios en los que se descompuso fue variable.

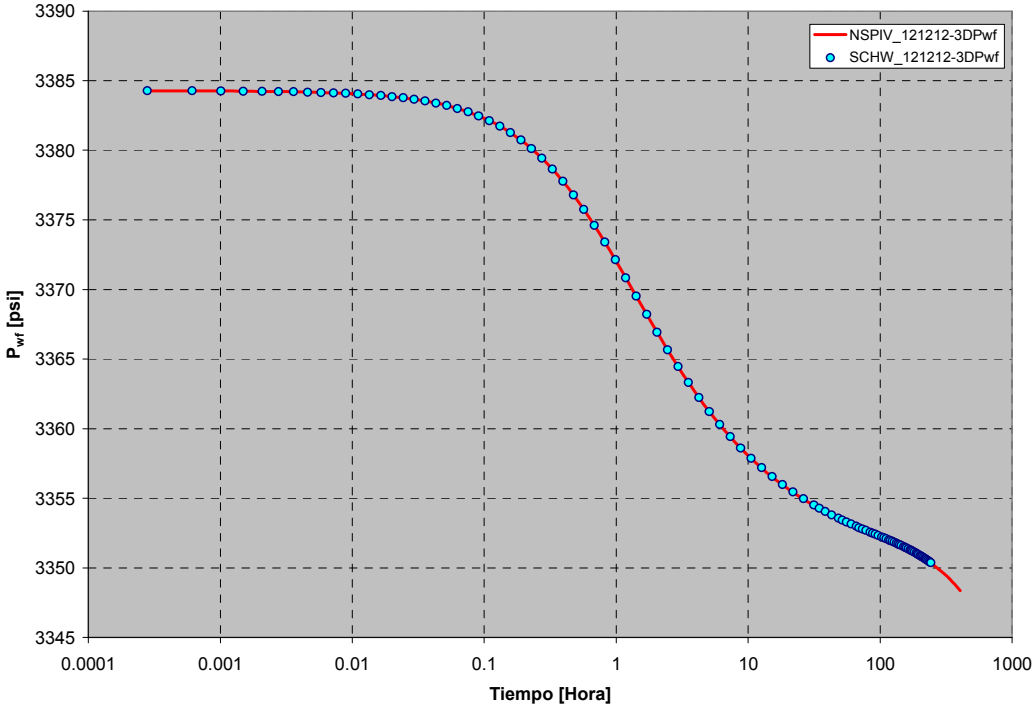
Con la restricción mencionada, el simulador original sólo resolvió hasta un sistema de 10x10x10. Pero, el simulador con Descomposición de Dominio, dividiendo el sistema en 4 subsistemas, y resolviendo con NSPIV (con la misma restricción) fue posible resolver, correctamente, hasta un sistema de 13x13x13. Para resolver sistemas más grandes a este último era necesario dividir el sistema global en más subsistemas, por lo tanto, para el caso de 14x14x14 se decidió descomponer en 8 subsistemas. Pero, este caso no fue resuelto de forma muy satisfactoria, ya que, la solución presentó diferencias con respecto a la real –

simulador original sin restricción-. El análisis de estas diferencias serán tratadas en las secciones 4.3.3 y 4.3.4.

Los datos empleados en la simulación son mostrados en la **tabla 4.11**, y los casos evaluados son presentados en la **tabla 4.12**. En las **figuras 4.17.a – 4.17.d**, se presentan las soluciones para P_{wf} con el simulador original (NSPIV, solución real) y con el simulador con Descomposición de Dominio (algoritmo Schwarz-NSPIV) para los diferentes casos. Como se puede observar en la **tabla 4.12**, y recordando que la subrutina de NSPIV fue limitada, empleando la Descomposición de Dominio fue posible resolver sistemas de ecuaciones más grandes que empleando sólo la solución directa, además, se obtuvieron soluciones bastante satisfactorias con el método propuesto para los casos desde 11x11x11 hasta 13x13x13, al igual que para los casos inferiores a 11x11x11. Sin embargo, para un sistema de 2744 ecuaciones con igual número de incógnitas (Modelo 3D de 14x14x14) se presentaron diferencias notables, lo cual no fue satisfactorio.

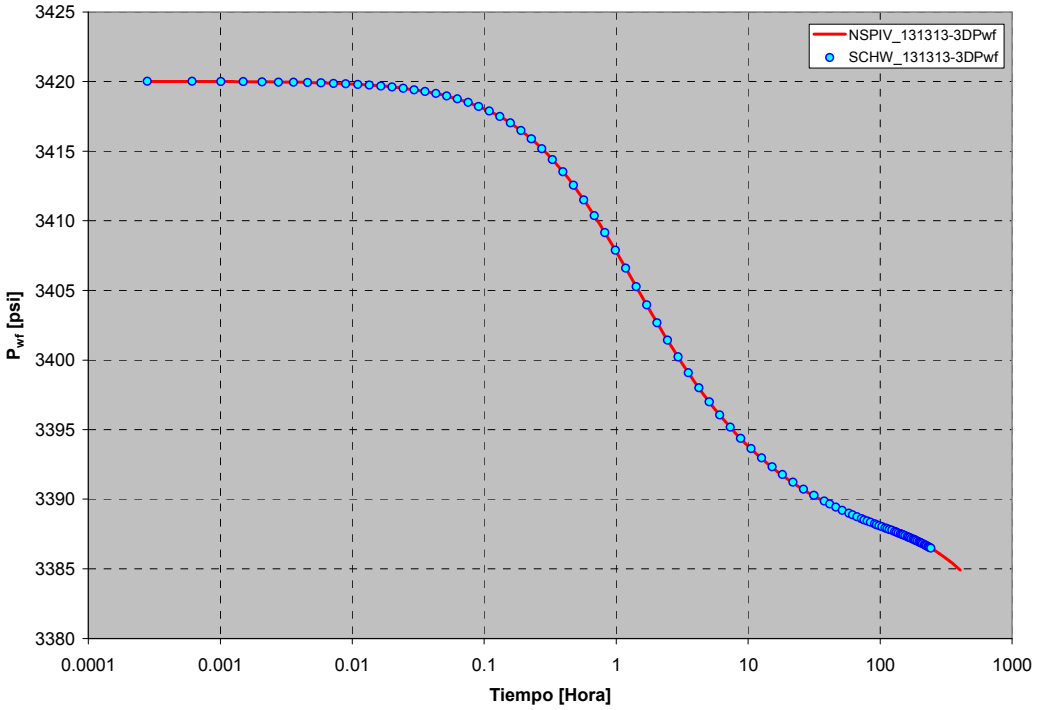


a) Solución para Pwf de un modelo de 11x11x11 aplicando Schwarz con 4 subsistemas.

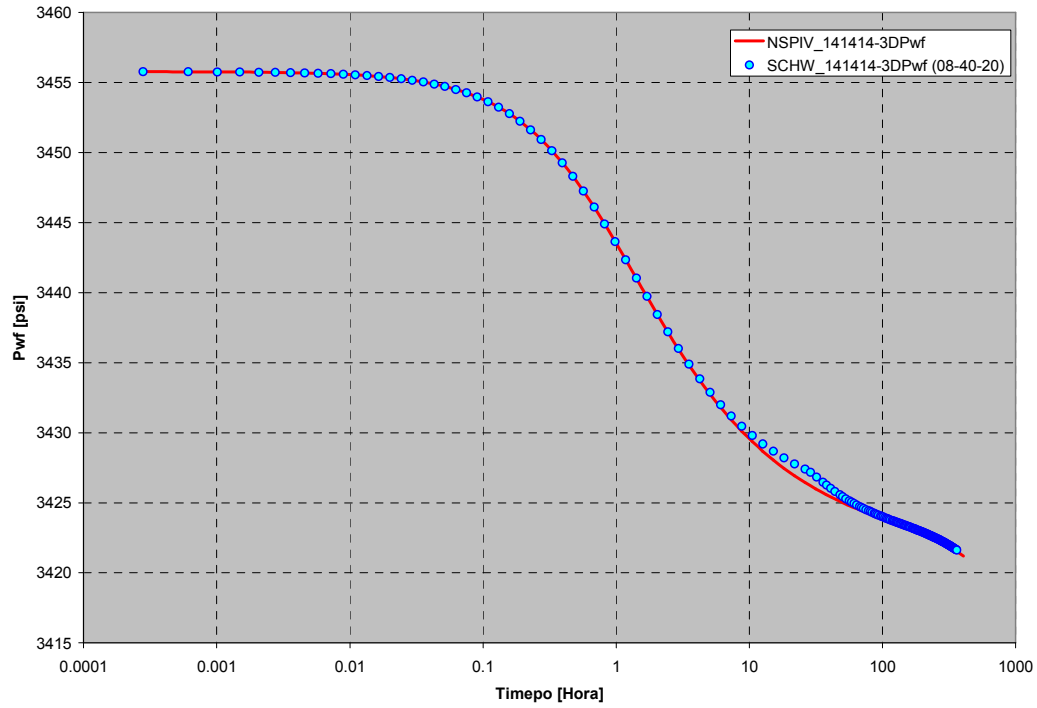


b) Solución para pwf de un modelo de 12x12x12 aplicando Schwarz con 4 subsistemas.

Figura 4.17. Gráfica de Pwf vs. tiempo mostrando la solución real obtenida con NSPIV y aplicando el algoritmo de Schwarz para mallas numéricas de diferentes tamaños.



c) Solución para pwf de un modelo de 13x13x13 aplicando Schwarz con 4 subsistemas.



d) Solución para Pwf de un modelo de 14x14x14. aplicando Schwarz con 8 subsistemas. Especialmente para este modelo se presentan diferencias entre la solución real y la solución aplicando el algoritmo de Schwarz.

Figura 4.17. Gráfica de Pwf vs. tiempo mostrando la solución real obtenida con NSPIV y aplicando el algoritmo de Schwarz para mallas numéricas de diferentes tamaños.

Tabla 4.11. Datos del yacimiento, fluido y pozo productor empleados en la simulación numérica.

Datos Generales del problema			
L_X	: $100.0*(N_x)^*$	μ	: 0.5
L_Y	: $100.0*(N_y)^*$	API	: 40.0
L_Z	: $100.0*(N_z)^*$	C_f	: 6.9E-06
N_x	: N_x^*	C_r	: 1.0E-06
N_y	: N_y^*	ϕ	: 0.15
N_z	: N_z^*	Bo	: 1.1
N_{WP}	: 1	P_r	: 500.0
N_{WI}	: 0	P_i	: 3000.0
T_{sim}	: 15.0		
K_X	: 10.0		
K_Y	: 10.0		
K_Z	: 10.0		

Datos del pozo productor	
X_P	: 1
Y_P	: 1
Z_P	: 1
r_{wp}	: 0.45
Q_o	: 100.0

* Los valores de *NX, Ny* y *Nz*, son tomados para cada uno de los casos de la tabla 4.12.

Tabla 4.12. Casos evaluados para mostrar el potencial del algoritmo de Schwarz.

N_x	N_y	N_z	N_c	Solución Directa.	Descomposición en 4 dominios.	Descomposición en 8 dominios.
9	9	9	729	X		
10	10	10	1000	X		
11	11	11	1331		X	
12	12	12	1728		X	
13	13	13	2197		X	
14	14	14	2744			X

4.3.3) Influencia de la localización del pozo en la exactitud de la solución.

Como se vio en el análisis anterior, al correr con Descomposición de Dominio, el caso de una malla de 14x14x14 en el que se dividía el sistema global en ocho subsistemas y el pozo se encontraba en la celda 1,1,1 (esquina frontal inferior izquierda de la malla), se observó que existía una separación o divergencia (**figura 4.17.d**), a un determinado tiempo, entre la solución real (solución directa, NSPIV) y la solución obtenida con Descomposición de Dominio (Schwarz-NSPIV).

Para poder saber la razón de esta separación temporal entre ambas soluciones, se corrieron varios casos para ver qué factor o factores tenían más influencia sobre la exactitud de la solución. Uno de los varios casos que se corrieron consistió en cambiar la posición del pozo; en vez de dejarlo en la celda 1,1,1 se colocó en la celda 7,7,7 (que se encuentra casi en el centro de la malla) mientras los demás datos permanecieron constantes.

En los resultados obtenidos se observó que sí tiene influencia la localización del pozo en la convergencia de la solución, pues, al colocarlo en el centro, como se puede ver en la **figura 4.19**, la divergencia disminuyó considerablemente (casi se eliminó) con respecto a la que existía en el caso en que el pozo estaba en la esquina. Esto se puede deber a que, en la esquina, la información se ve más afectada por los efectos del transiente y en el centro no, debido a que en la esquina, por ser el límite del yacimiento, no existen transmisibilidades en $i_{-1/2}$, $j_{-1/2}$ y $k_{-1/2}$, y en el centro o en cualquier otra celda, que no se encuentre en la orilla, sí existen. Por lo tanto, se debe procurar que el pozo o pozos que van a estar en la simulación queden lo más centrados posible o, al menos, que no estén en las orillas o muy cerca de éstas, con el fin de evitar la diferencia que se tiene con Schwarz al tener el pozo o pozos en las orillas y, así obtener resultados más exactos.

Las **figuras 4.18** y **4.19** muestran la comparación de resultados entre la solución directa y la solución con Descomposición de Dominio para el caso en que el pozo se colocó en la celda 1,1,1 y en la celda 7,7,7, respectivamente. La **tabla 4.13** muestra los datos usados en las simulaciones de este análisis.

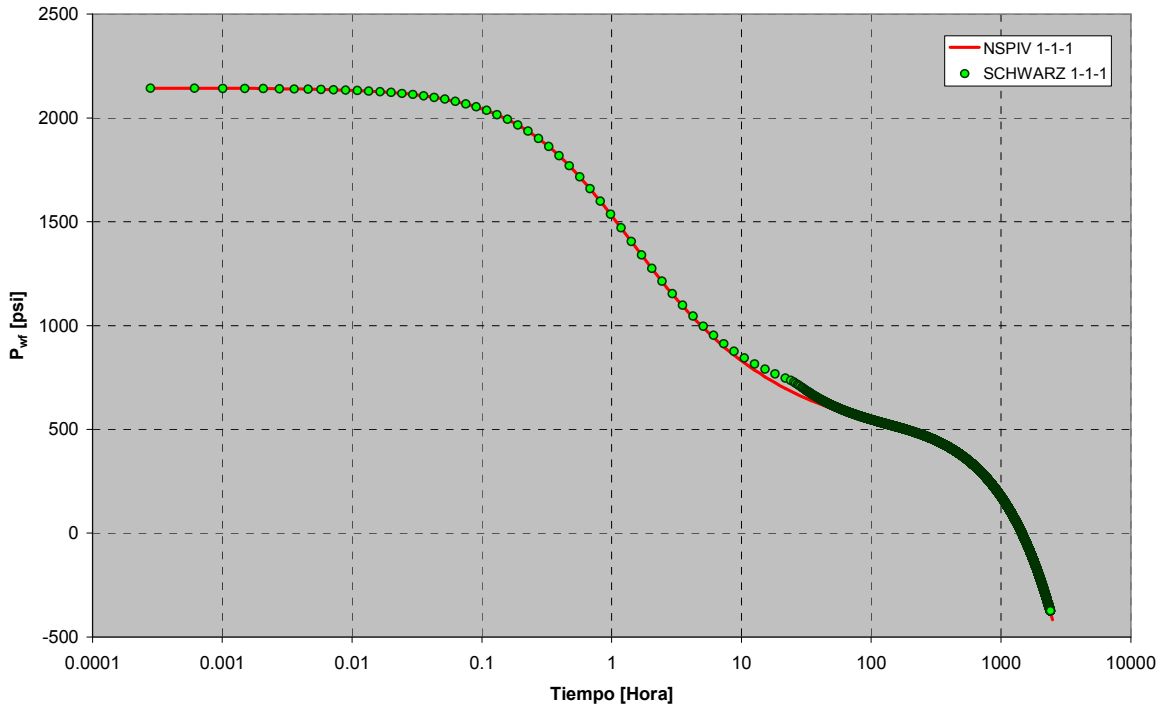


Figura 4.18. Donde se muestra la divergencia que existe, a un tiempo determinado, entre Schwarz y NSPIV en el caso en que el pozo se encuentra en la celda 1,1,1.

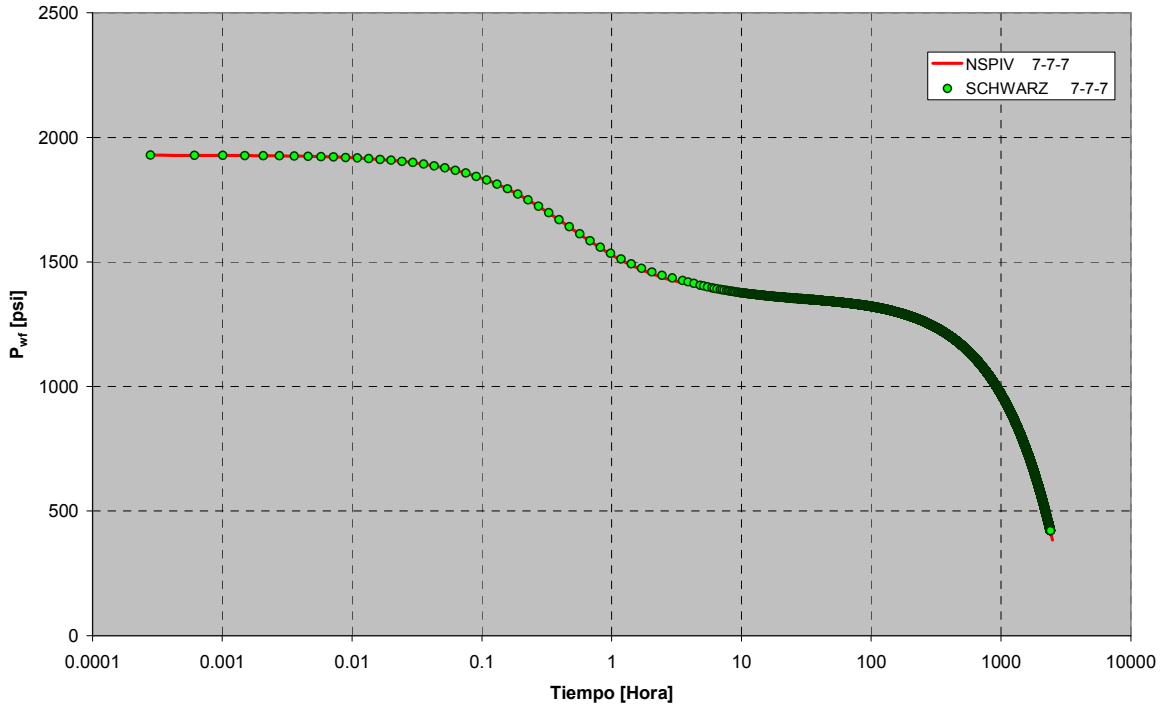


Figura 4.19. Donde se muestra la reducción de la divergencia al localizar el pozo en la celda 7-7-7 (la cual esta cerca al centro de la malla).

Tabla 4.13. Datos del yacimiento, fluido y pozo productor empleados en la simulación numérica para el caso del análisis de la localización del pozo.

Datos Generales del problema			
L_X	: 1400.0	μ	: 0.5
L_Y	: 1400.0	API	: 40.0
L_Z	: 1400.0	C_f	: 6.9E-06
N_x	: 14	C_r	: 1.0E-06
N_y	: 14	ϕ	: 0.15
N_z	: 14	Bo	: 1.1
N_{WP}	: 1	P_r	: 500.0
N_{WI}	: 0	P_i	: 3000.0
T_{sim}	: 100.0		
K_X	: 10.0		
K_Y	: 10.0		
K_Z	: 10.0		
Datos del pozo productor			
	Caso 1	**Caso 2**	
X_P	: 1	7	
Y_P	: 1	7	
Z_P	: 1	7	
r_{wp}	: 0.45	0.45	
Qo	: 100.0	100.0	

4.3.4) Influencia del Número de Subsistemas, del Traslape y del Número Máximo de Iteraciones sobre la solución y el tiempo de ejecución.

Otro de los análisis que se hicieron para saber porqué existía una separación temporal entre la solución real y la solución con Descomposición de Dominio, para el caso de una malla de 14x14x14 subdividida en ocho subdominios y con el pozo en la celda 1,1,1, consistió en saber qué influencia tenían los siguientes parámetros: número de subsistemas en que se divide el sistema global, el traslape que existe entre éstos y, el número de iteraciones máximas en las que la subrutina Schwarz debe llegar a la solución deseada del sistema de ecuaciones global antes de disminuir el tamaño del incremento de tiempo (Δt).

Los diferentes casos que se corrieron en este análisis fueron tomados de las diferentes combinaciones posibles que se podían tener al; 1) variar el número de subsistemas, en que se dividía el sistema global, entre 6, 7, 8, 9 y 10 subsistemas; 2) variar el tamaño del traslape de los subsistemas entre 20, 40 y 60 y; 3) variar el número de iteraciones máximas permisibles en Schwarz entre 5, 10 y 20 iteraciones.

La **tabla 4.14** muestra los datos empleados en las simulaciones de este análisis.

Los resultados, más representativos, obtenidos de todas las combinaciones posibles de estos casos, comparados con la solución directa (NSPIV), son mostrados de la **figura 4.20.a – 4.20.c**.

De los resultados obtenidos y mostrados en las **figuras 4.20.a – 4.20.c.**, se puede observar que el parámetro que más afecta a la exactitud de la solución es el número de subdominios en que se divide el dominio total, es decir, el número de subsistemas (submatrices) en las cuales se subdivide el sistema de ecuaciones global. Por lo tanto, mientras menor sea el número de subsistemas en que se divide el sistema de ecuaciones global, mayor es la exactitud de la solución y viceversa. Esto se debe a que, mientras menor es el número en que se subdivide el sistema de ecuaciones total, más grandes son los subsistemas generados y, por lo tanto, la pérdida de información es menor, pues cada uno de ellos tiene mayor alcance (reciben más información). En cambio, en el caso en que se divide el sistema global en muchos, éstos tienen un menor tamaño y por eso la pérdida de información que se tiene es mayor, causando una menor exactitud. De hecho, en la descomposición del dominio matemático siempre se va a tener pérdida de información, pues los subsistemas no alcanzarán a cubrir todas las zonas del sistema global. El único caso en que se lograría abarcar toda la información sería en el que éste se dividiera en un solo subsistema, pero eso lógicamente no es Descomposición de Dominio, pues no se estaría descomponiendo el sistema de ecuaciones original, sino, resolviendo íntegramente.

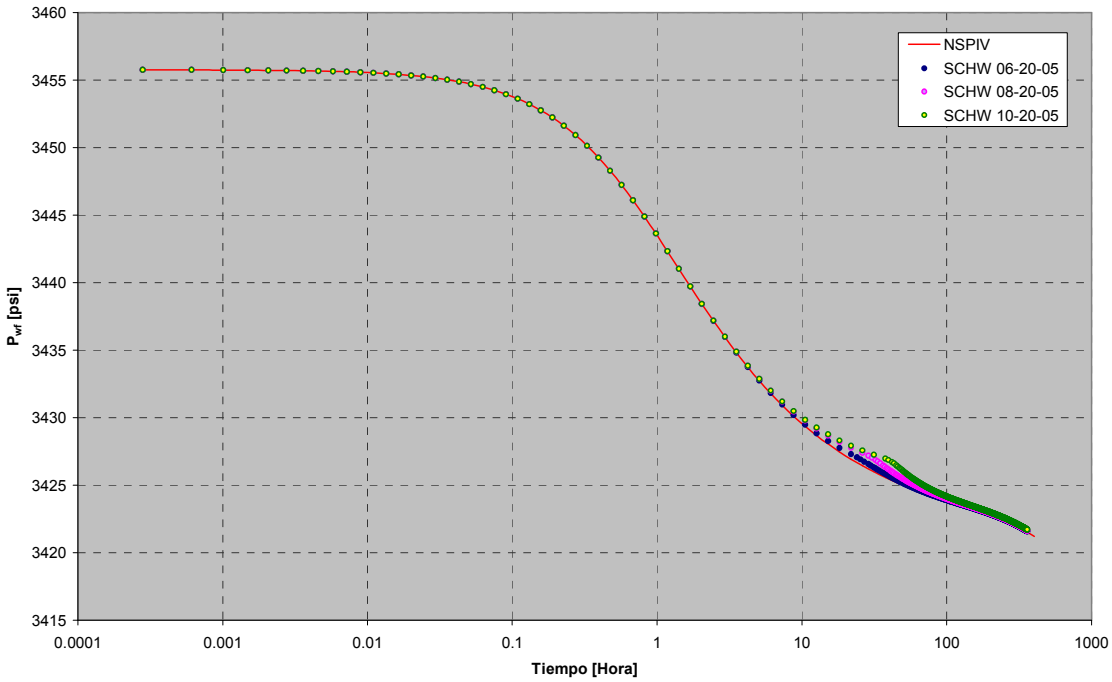
El segundo parámetro que más afecta en la exactitud de la solución, como se puede ver, es el tamaño de traslape. Mientras mayor sea el traslape, mayor es la exactitud de la solución.

Y esto es lógico, pues el traslape ayuda a aumentar el tamaño de los subsistemas. Para comprender esto se debe recordar que, el traslape nos dice cuánta información están compartiendo entre sí los subsistemas que dividen el sistema global, y esto se debe a que dicho sistema global no es dividido en subsistemas aislados, sino, en subsistemas que se traslapan entre sí y que, por lo tanto, tienen zonas en donde comparten información con los otros. De esta forma, mientras mayor sea el traslape, mayor es el tamaño de los subsistemas, pues tienen que cumplir con el requisito de traslape pedido. Sin embargo, a pesar de que este parámetro de traslape nos ayuda a colectar más información del sistema total, su impacto en la exactitud no es tan fuerte como el del parámetro de número de subsistemas, pues, aunque con el traslape se aumenta el tamaño de los subsistemas éste no se incrementa tanto como cuando se subdivide el sistema real en una menor cantidad de subsistemas.

Por último, el parámetro que menos influye, de hecho casi ni lo hace, en la exactitud de la solución es el número de iteraciones. Sin embargo, aunque no tuvo efecto en la confiabilidad de los resultados sí tuvo influencia en el tiempo de ejecución del programa. Las **figuras 4.21.a. – 4.21.c** muestran los resultados obtenidos en el tiempo de corrida para diferentes esquemas: variando el número de subdominios, el traslape y el número de de iteraciones máximas. Se observó que con el esquema de 20 iteraciones máximas el tiempo de ejecución era mucho menor que con los otros dos esquemas de 5 y 10 iteraciones máximas.

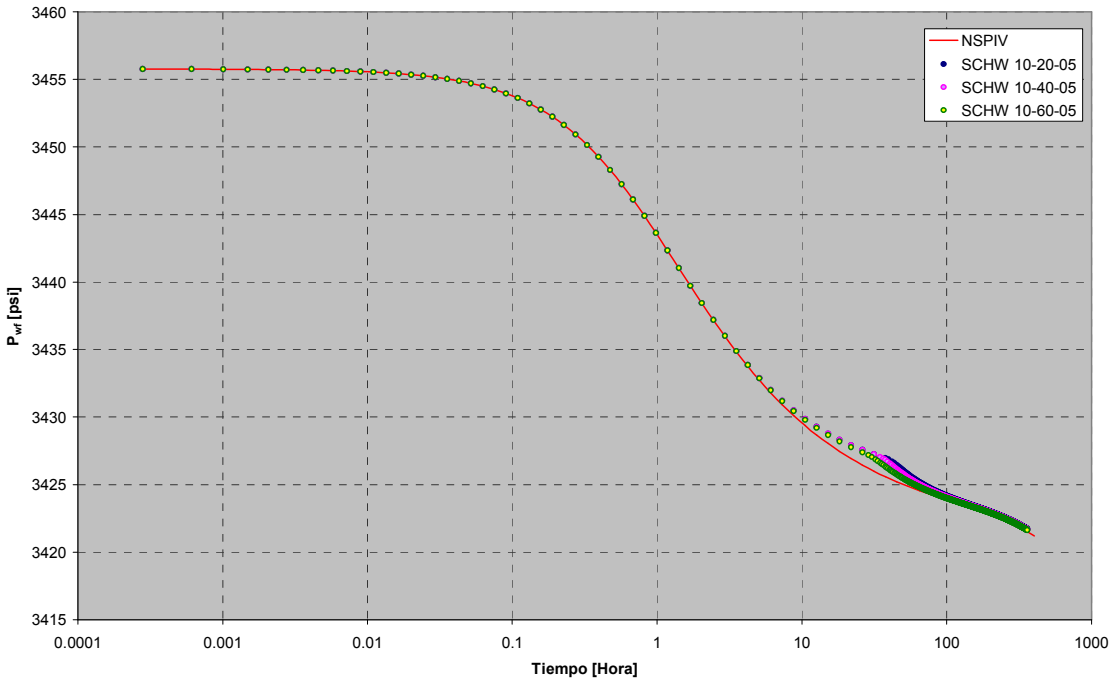
De las observaciones anteriores se puede decir que el parámetro que más influye en la exactitud de la solución es el número de subsistemas en los que se divide el sistema de ecuaciones global y el otro parámetro que influye en la exactitud, aunque no con tanta fuerza, es el tamaño del traslape que existe entre los subsistemas, mientras que el número de iteraciones máximas no influyó en la confiabilidad de los resultados. Por otro lado, se observó que si el número de subdominios y el número de iteraciones máximas crecen, el tiempo de corrida se reduce, mientras que el incremento en el tamaño de traslape aumenta el tiempo de corrida. Por lo tanto, el programador o el usuario debe jugar con estos

parámetros hasta encontrar la combinación que más le convenga; una combinación que permita obtener buenos resultados y en poco tiempo.

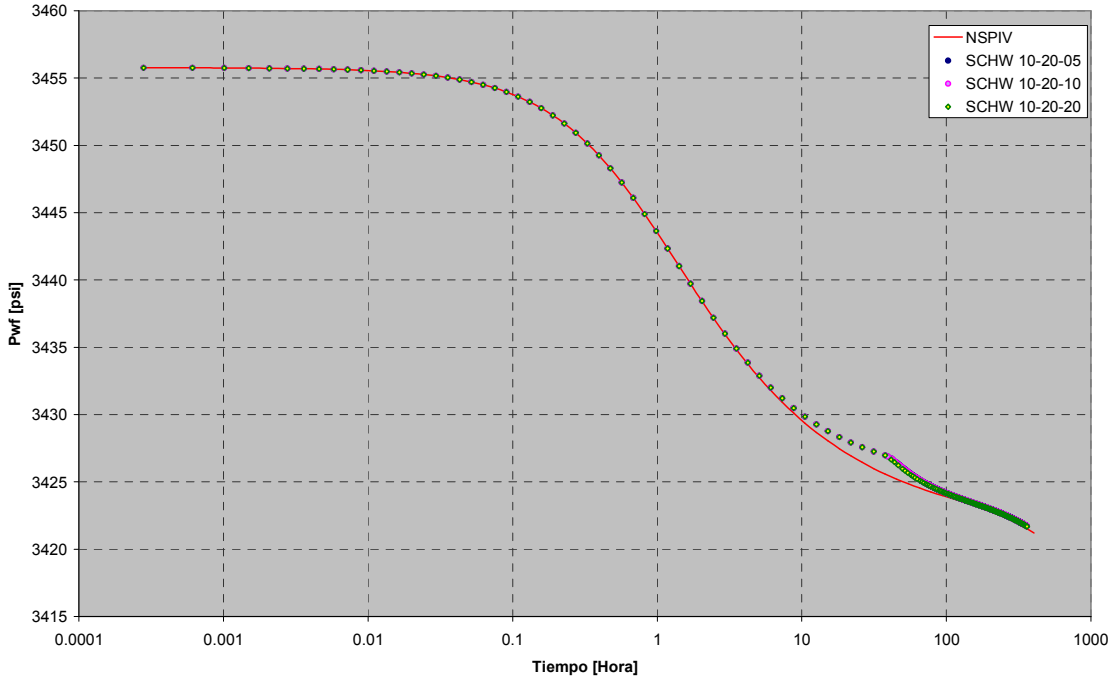


a) Influencia del número de subdominios, sobre la exactitud de la solución, al usar 6, 8 y 10 subdominios, y mantener constantes el tamaño de traslape en 20 y el número de iteraciones en 5.

Figura 4.20. Influencia del número de subdominios, numero de iteraciones máximas y traslape en la exactitud de la solución.

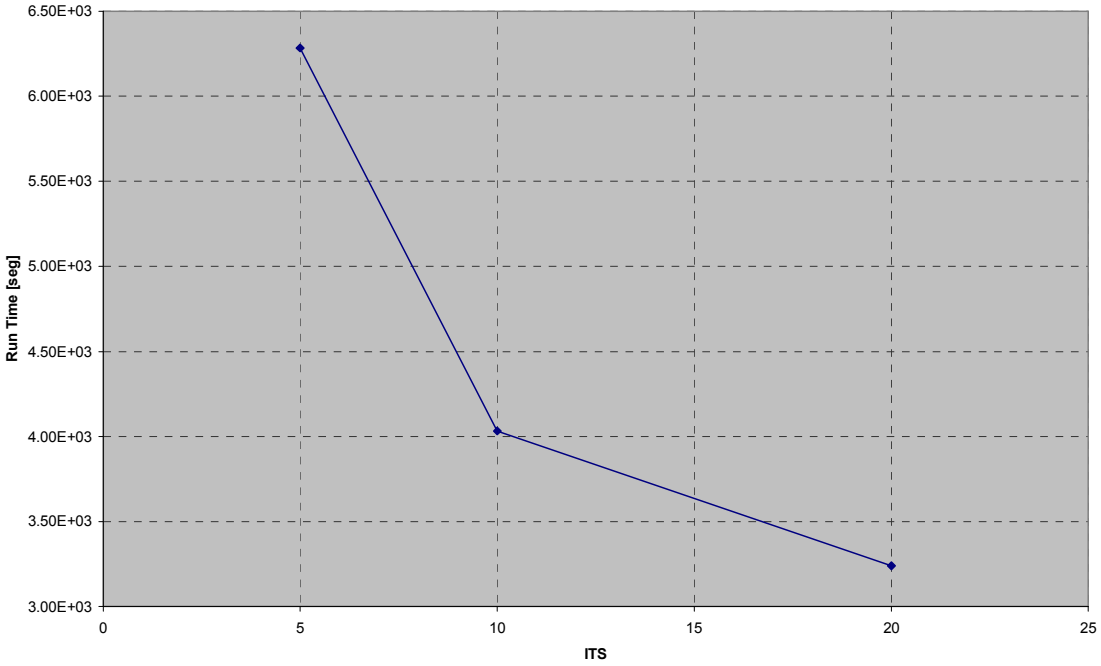


b) Influencia del tamaño del traslape , en la exactitud de la solución, al usar 20, 40 y 60 de tamaño de traslape, y mantener constantes el número de subdominios en 10 y el número de iteraciones en 5.

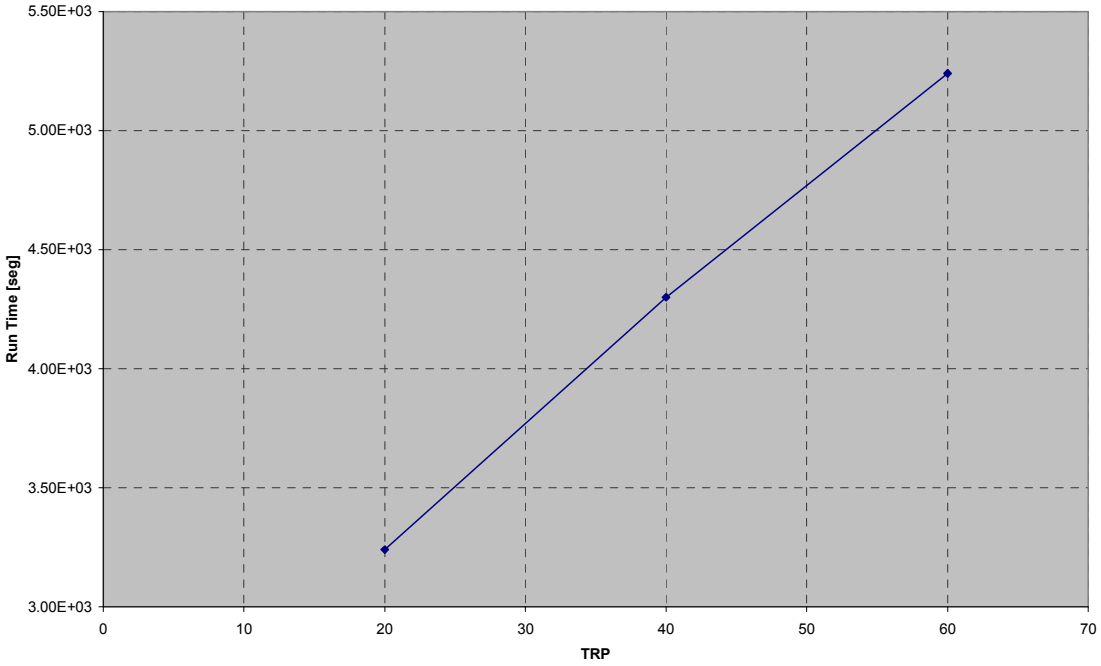


c) Influencia del número de iteraciones, en la exactitud de la solución, al usar 05, 10 y 20 iteraciones máximas, y mantener constantes el número de subdominios en 10 y el tamaño de traslape en 20.

Figura 4.20. Influencia del numero de subdominios, numero de iteraciones máximas y traslape en la exactitud de la solución.

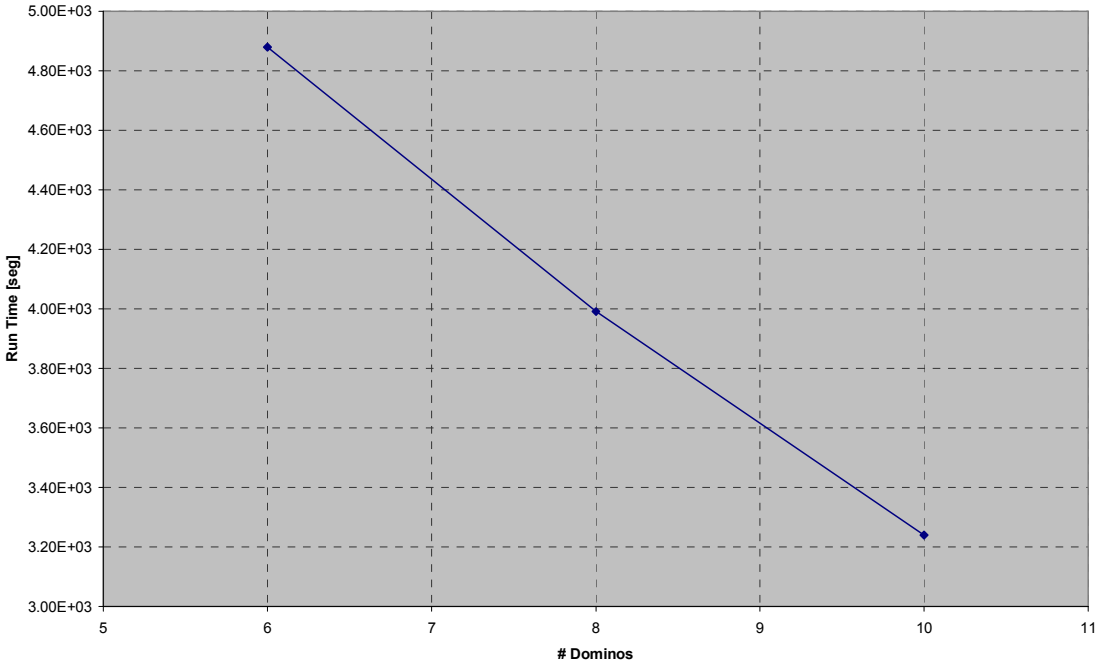


a) Influencia del número de iteraciones, en el tiempo de corrida,, al usar 05, 10 y 20 iteraciones máximas, y mantener constantes el número de subdominios en 10 y el tamaño de traslape en 20.



b) Influencia del tamaño del traslape, en el tiempo de corrida, al usar 20, 40 y 60 de tamaño de traslape, y mantener constantes el número de subdominios en 10 y el número de iteraciones en 5.

Figura 4.21. Influencia del número de subdominios, numero de iteraciones máximas y traslape en el tiempo de corrida.



c) Influencia del número de subdominios, en el tiempo de corrida, al usar 6, 8 y 10 subdominios, y mantener constantes el tamaño de traslape en 20 y el número de iteraciones en 5.

Figura 4.21. Influencia del número de subdominios, numero de iteraciones máximas y traslape en el tiempo de corrida.

Tabla 4.14. Datos del yacimiento, fluido y pozo productor empleados en la simulación numérica para el caso del análisis de la localización del pozo.

Datos Generales del problema			
L_X	: 1400.0	μ	: 0.5
L_Y	: 1400.0	API	: 40.0
L_Z	: 1400.0	C_f	: 6.9E-06
N_x	: 14	C_r	: 1.0E-06
N_y	: 14	ϕ	: 0.15
N_z	: 14	Bo	: 1.1
N_{WP}	: 1	P_r	: 500.0
N_{WI}	: 0	P_i	: 3000.0
T_{sim}	: 15.0		
K_X	: 10.0		
K_Y	: 10.0		
K_Z	: 10.0		
Datos del pozo productor			
X_P	: 1		
Y_P	: 1		
Z_P	: 1		
r_{wp}	: 0.45		
Qo	: 100.0		

4.4) Paralelización del Simulador de Descomposición de Dominio sobre el Sistema de Ecuaciones Lineales.

Una vez que el simulador con Descomposición de Dominio estaba dando buenos resultados y teniendo un buen rendimiento, se procedió a su paralelización. El propósito de ésta era optimizar la ejecución del programa disminuyendo su tiempo de corrida.

Para paralelizar se utilizó la librería MPI, que es una librería estándar de envío de mensajes. Se utilizó esta librería debido a que estaba disponible, es fácil de manejar y se tenían algunos manuales que facilitaron su rápido entendimiento. Por otra parte, el envío de mensajes es el esquema de programación paralela que ha cobrado mayor importancia en los últimos años.

La primera etapa de la paralelización consistió en hacer un esquema general de la subrutina de Schwarz con el fin de detectar las zonas o procesos en donde se podría paralelizar, es decir, puntos del código en donde al distribuirse el trabajo entre procesadores la ejecución del programa sería optimizada.

En la segunda etapa se elaboró un esquema global del simulador paralelizado, o sea, un esquema en donde se muestran los lugares en donde los procesadores recibirían sus datos y en donde compartirían sus resultados con otros. Esto se hizo para que el proceso de paralelización fuera más fácil y más rápido, pues, siempre es bueno analizar el problema antes de atacarlo. En la **figura 4.22.** se muestran los esquemas para el programa principal y para la rutina de Schwarz.

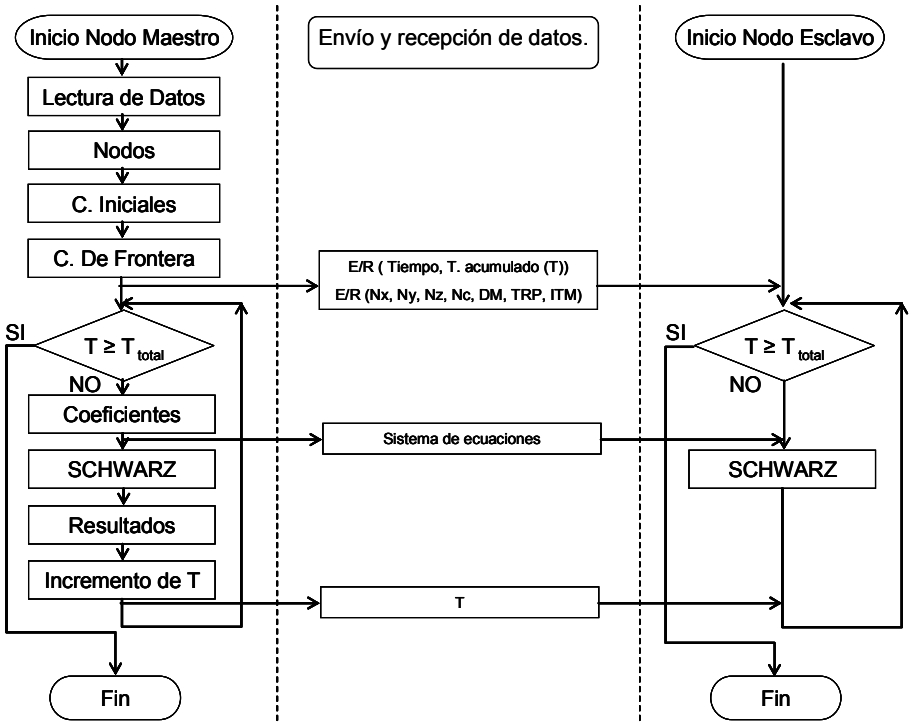
Se corrió el simulador en su forma paralela para comparar la exactitud de sus resultados contra los resultados del simulador secuencial, por medio de gráficas de P_{wf} contra tiempo, así como la diferencia entre los tiempos de ejecución de ambos. La **figura 4.23** muestra la comparación de los resultados entre el simulador paralelo de Descomposición de Dominio y el simulador secuencial de Descomposición de Dominio.

En la **figura 4.24** se muestra la comparación de los tiempos de ejecución entre el simulador secuencial y el simulador paralelo utilizando varios procesadores. La **figura 4.25** muestra con mayor detalle la diferencia en tiempo de ejecución que existe cuando se usa distinto número de procesadores en la ejecución del programa paralelo.

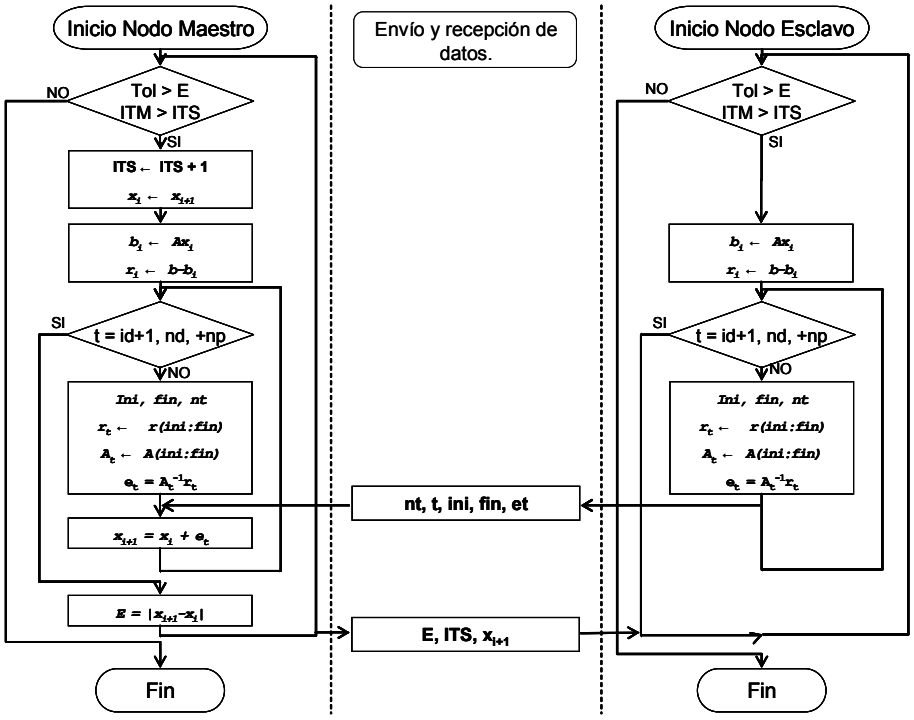
Como se puede ver, en la **figura 4.23**, los resultados fueron idénticos a los obtenidos con el simulador secuencial, lo cual indica que la paralelización no provocó errores.

Por otra parte, en la **figura 4.24** se puede apreciar que con la paralelización el tiempo de ejecución realmente se optimizó. El simulador en su forma paralela, utilizando solo dos procesadores, se tarda alrededor de la décima parte de lo que se tarda el simulador en su forma secuencial. Además, se puede apreciar que si se utilizan más procesadores el tiempo de ejecución disminuye aun más. Esto es importante, pues si el simulador tardara doce horas en entregar resultados, con este nuevo esquema paralelo –usando sólo dos procesadores- tardaría aproximadamente una hora con doce minutos.

Lo anterior demuestra que la Programación en Paralelo, para este caso de Descomposición de Dominio, realmente ayuda a optimizar la ejecución del programa sin afectar la exactitud de sus resultados.



a) Programa principal.



b) Subrutina Schwarz.

Figura 4.22. Diagrama de flujo del esquema paralelo.

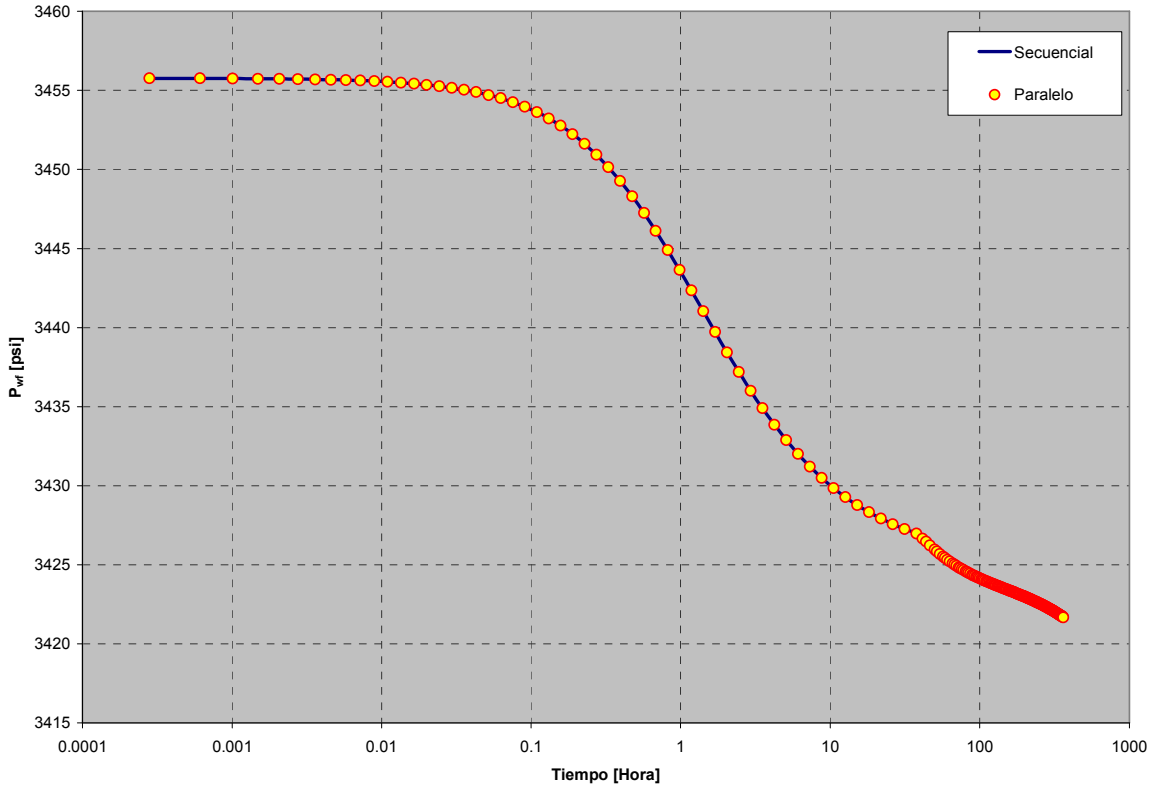


Figura 4.23. Comparativo de Pwf entre los dos programas, secuencial y paralelo, de Descomposición de Dominio.

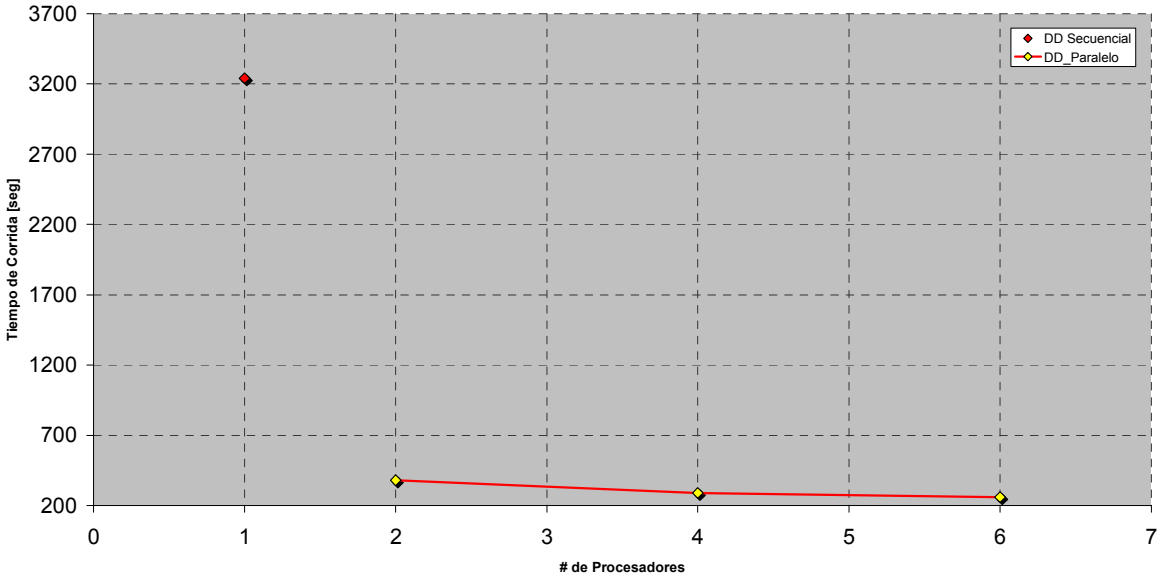


Figura 4.24. Diferencias de tiempo entre los dos programas, secuencial y paralelo, de Descomposición de Dominio.

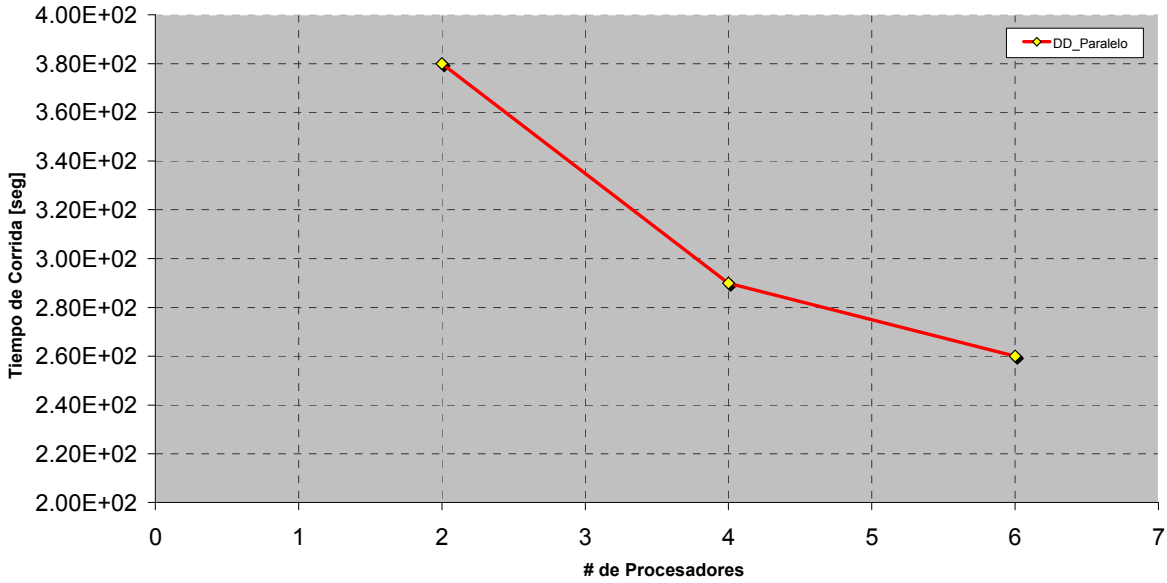


Figura 4.25. Diferencia de tiempo para el programa paralelo de Descomposición de Dominio con diferente número de procesadores.

4.5) Comentarios finales del Capítulo.

Se ha expuesto en este capítulo parte del trabajo realizado para implementar la Descomposición de Dominio en el simulador y para ejecutarlo en forma paralela. Claro está que el verdadero trabajo fue mayor –se hicieron muchos análisis y mejoramientos al programa-, pero, como gran parte del trabajo se basó principalmente en mejorar la manera en como son manejados los datos y las operaciones realizadas en el programa, los análisis y trabajos realizados en estos aspectos no son mostrados pues no contribuyen con el propósito central de esta tesis: probar que la Descomposición de Dominio puede ser aplicada a la SNY con buenos resultados y que se puede utilizar la Programación en Paralelo para mejorar su desempeño.

Sin embargo, con los análisis que se mostraron en este capítulo se puede decir que:

1. La descomposición del dominio físico es altamente efectiva cuando se manejan solo dos subdominios, al grado de que los resultados, tanto en la distribución de presión

como en el comportamiento de P_{wf} son prácticamente idénticos. Por otro lado, aunque la descomposición del dominio matemático permite dividir el sistema global en un mayor número de subsistemas (subdominios), su exactitud y tiempo de corrida están fuertemente influenciadas por el número de subsistemas, el tamaño del traslape y el número máximo de iteraciones permitidas, lo cual requiere de un esfuerzo adicional para determinar los parámetros óptimos. Sin embargo, se observó que se tienen mejores resultados en los casos donde la dimensión de cada subsistema representa al menos un 17% del tamaño del sistema global y con un traslape de entre 10 y 15% del tamaño del subsistema. Así mismo, se observó que un número de máximo de veinte iteraciones en Schwarz permitió obtener los mejores tiempos en todos los casos.

2. La Programación en Paralelo realmente contribuyó en la reducción del tiempo de ejecución, observándose que el tiempo de corrida, para este caso en particular, se redujo aproximadamente una décima parte del tiempo requerido con el programa secuencial.

3. Aunque el análisis en detalle no es mostrado en este trabajo, es importante destacar que el programa de Descomposición de Dominio aun con Programación en Paralelo es un poco más lento, en relación con el programa original (sin Descomposición de Dominio ni Programación en Paralelo); para el caso específico del modelo en 3D de 14x14x14, el simulador con Descomposición de Dominio y Programación en Paralelo con dos procesadores requiere de alrededor de 100 segundos más que el simulador que no posee estas características. Aun así, se debe recordar que la capacidad de cómputo se incrementó, de tal forma, que incluso cuando se requiere un mayor tiempo de ejecución es posible resolver sistemas de mayor dimensión con el simulador con Descomposición de Dominio y Programación en Paralelo.

CAPÍTULO V. APLICACIONES.

En este capítulo se mostrarán tres casos de simulación con diferente tamaño de malla, para comparar el desempeño del simulador con Descomposición de Dominio y Programación en Paralelo (DD-PP) contra el simulador secuencial con Descomposición de Dominio (DD-PS) y contra el simulador original (SO-PS) **(Hernández R, O. Y Del Valle M, P., 2005)**.

Para los tres casos se utilizó como parámetros de descomposición: una partición en seis subsistemas con un traslape de 60 y veinte iteraciones máximas para la subrutina Schwarz. Además, para el simulador con Programación en Paralelo se evaluó su rendimiento al trabajar con dos, cuatro y seis procesadores, excepto para el caso 3 en el cual sólo se utilizaron seis procesadores.

Cabe mencionar que, para los tres casos, los simuladores se corrieron en una misma máquina con varios procesadores (cluster), en la cual todos los procesadores son iguales y tienen las mismas características. Esto se hizo con la intención de eliminar las diferencias en el tiempo de corrida debidas a la ejecución del simulador en procesadores que tienen diferentes características entre sí y, de esta manera, conocer de manera precisa el impacto que tiene la Descomposición de Dominio y la Programación en Paralelo sobre la exactitud de los resultados y el tiempo de corrida en el simulador. Claro está, que para el simulador original (SO-PS) y el simulador secuencial con Descomposición de Dominio (DD-PS) se utilizó sólo un procesador del cluster para realizar las corridas y, para el simulador de Descomposición de Dominio y Programación en Paralelo (DD-PP) se utilizaron tantos procesadores como requería la corrida (2, 4 ó 6). Por último, vale la pena mencionar que para evitar que los simuladores interfirieran entre sí en su tiempo de corrida se ejecutó cada simulador por separado, es decir, no se corrieron dos simuladores simultáneamente; sólo uno corría en la máquina y una vez que éste concluía su corrida de simulación se ejecutaba otro simulador. De esta manera los tiempos medidos, con cada simulador, son confiables y no existe incertidumbre de que la ejecución de un simulador haya interferido en el tiempo de corrida de otro.

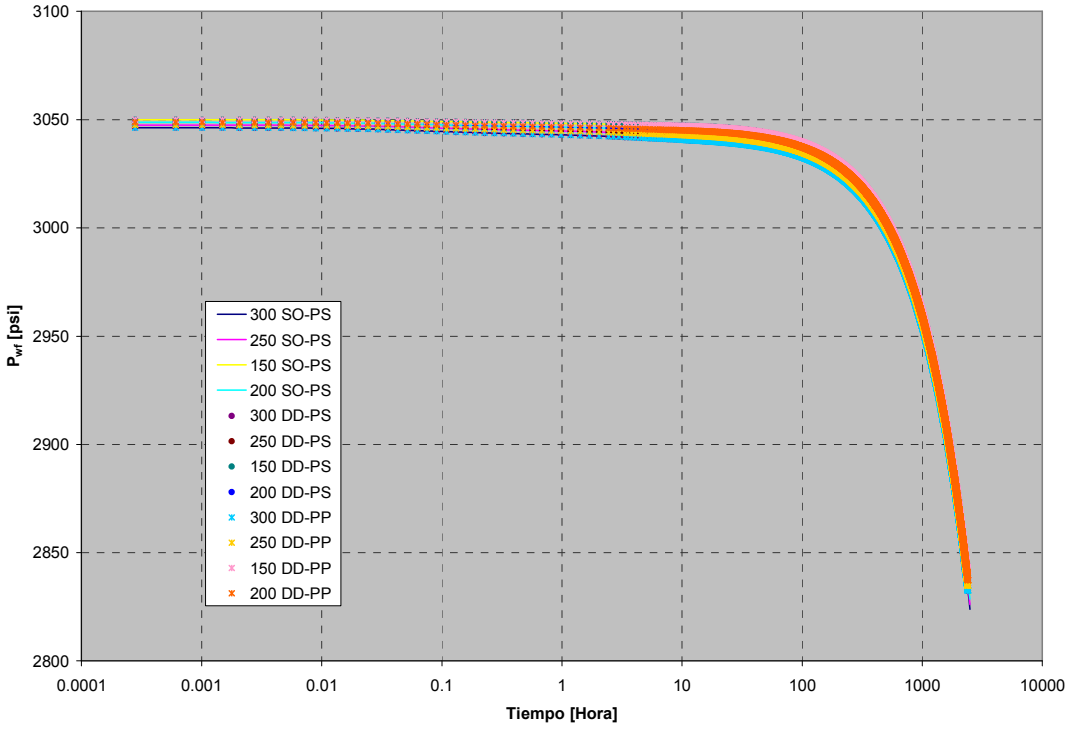
5.1) Caso 1.

Simulación de un arreglo de cinco pozos para una malla de 15x15x3:

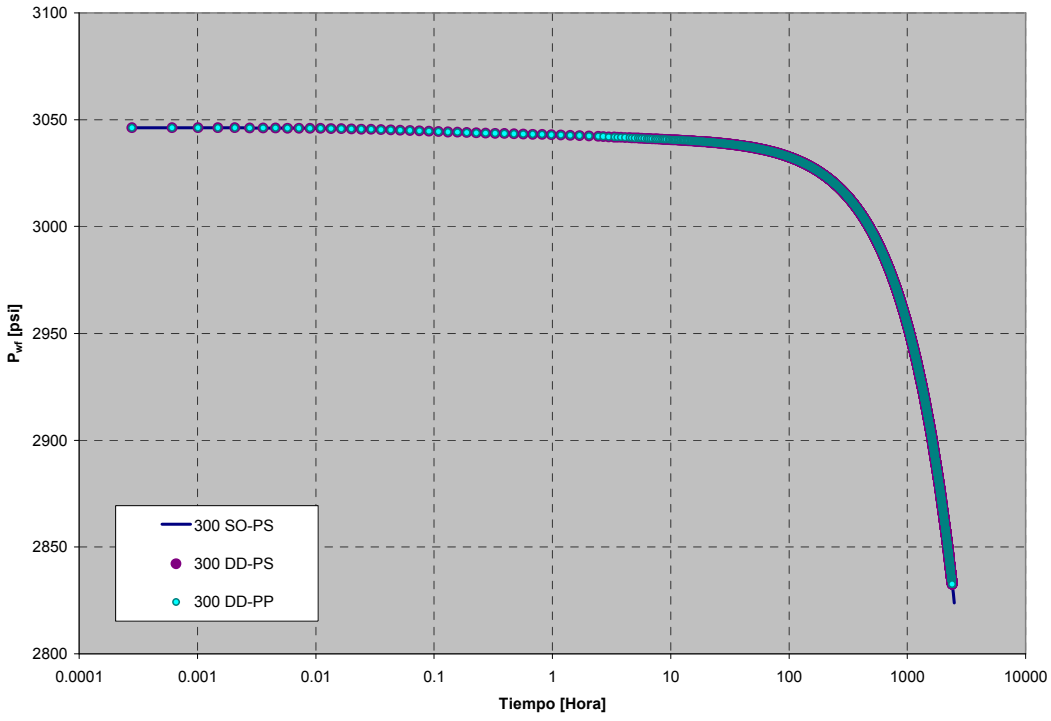
En este caso de aplicación se empleó un arreglo de cinco pozos sobre un yacimiento hipotético de 1500x1500x300 [pies^3]. Esta malla genera un sistema de ecuaciones lineales de 675 incógnitas. Los datos generales del yacimiento y de los pozos se presentan en las **tablas 5.1 y 5.2**, respectivamente**.**

En la **figura 5.1.a.** se muestra el comportamiento de los cuatro pozos productores obtenido con cada uno de los simuladores. Para el caso del simulador con Programación en Paralelo sólo se utilizan los resultados obtenidos al utilizar seis procesadores. Debido a que en este caso las curvas de los diferentes pozos están muy cercanas entre sí como para apreciar el buen ajuste de los resultados, en la **figura 5.1.b.** se muestra el comportamiento de un solo pozo (pozo1).

En la **figura 5.2** se da a conocer el tiempo de ejecución de los simuladores. Nótese que en esta gráfica los casos que indican un procesador corresponden al simulador original y al simulador con Descomposición de Dominio secuencial.



a) Comportamiento de los cuatro pozos productores.



b) Comportamiento del pozo productor 1.

Figura 5.1. Comportamiento de la Pwf contra el tiempo para el **caso 1**.

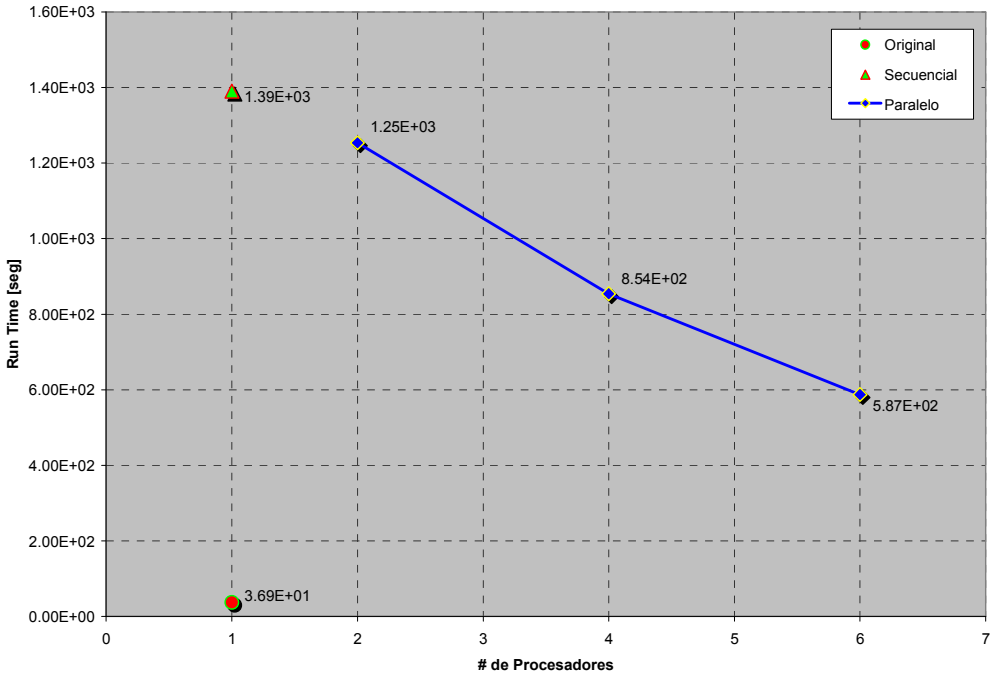


Figura 5.2. Tiempo de corrida (Run Time) registrado para el **caso 1** con los diferentes simuladores.

Tabla 5.1. Datos del yacimiento, fluido y pozo productor empleados en los diferentes simuladores para el **caso 1.**						
Datos Generales del problema						
L_X	:	1500.0		μ	:	0.920
L_Y	:	1500.0		API	:	40.0
L_Z	:	300.0		C_f	:	6.9E-06
N_x	:	15		C_r	:	1.0E-06
N_y	:	15		ϕ	:	0.15
N_z	:	3		Bo	:	1.1
N_{WP}	:	4		P_r	:	500.0
N_{WI}	:	1		P_i	:	3000.0
T_{sim}	:	100.0				
K_X	:	100.0				
K_Y	:	100.0				
K_Z	:	100.0				

TABLA 5.2. Datos de los pozos para el **caso 1.**

Pozos Productores					
Pozo	X_p	Y_p	Z_p	r_{wp}	Q_{op}
1	2	2	2	0.45	-150.0
2	2	14	2	0.45	-125.0
3	14	2	2	0.45	-75.0
4	14	14	2	0.45	-100.0
Pozos Inyectores					
Pozo	X_p	Y_p	Z_p	r_{wp}	Q_{op}
1	8	8	1	0.45	+175.0

5.2) Caso 2.

Simulación de un arreglo de cinco pozos para una malla de 30x30x5:

En este segundo caso se empleó un arreglo igual al del **caso 1**, de cinco pozos, pero con un sistema de mayor dimensión, un yacimiento hipotético de 3000x3000x500 [pies^3]. Esta malla genera un sistema de ecuaciones lineales de 4500 incógnitas. Los datos generales del yacimiento y de los pozos se presentan en las **tablas 5.3 y 5.4**, respectivamente**.**
En la **figura 5.3** se muestra el comportamiento de los cuatro pozos productores obtenido con cada uno de los simuladores. Para el caso de el simulador con Programación en Paralelo sólo se utilizan los resultados obtenidos al utilizar seis procesadores, ya que, los resultados no dependen del número de procesadores empleados.

En la **figura 5.4** se da a conocer el tiempo de ejecución de los simuladores. Nótese que en esta gráfica los casos que indican un procesador corresponden al simulador original y al simulador con Descomposición de Dominio secuencial.

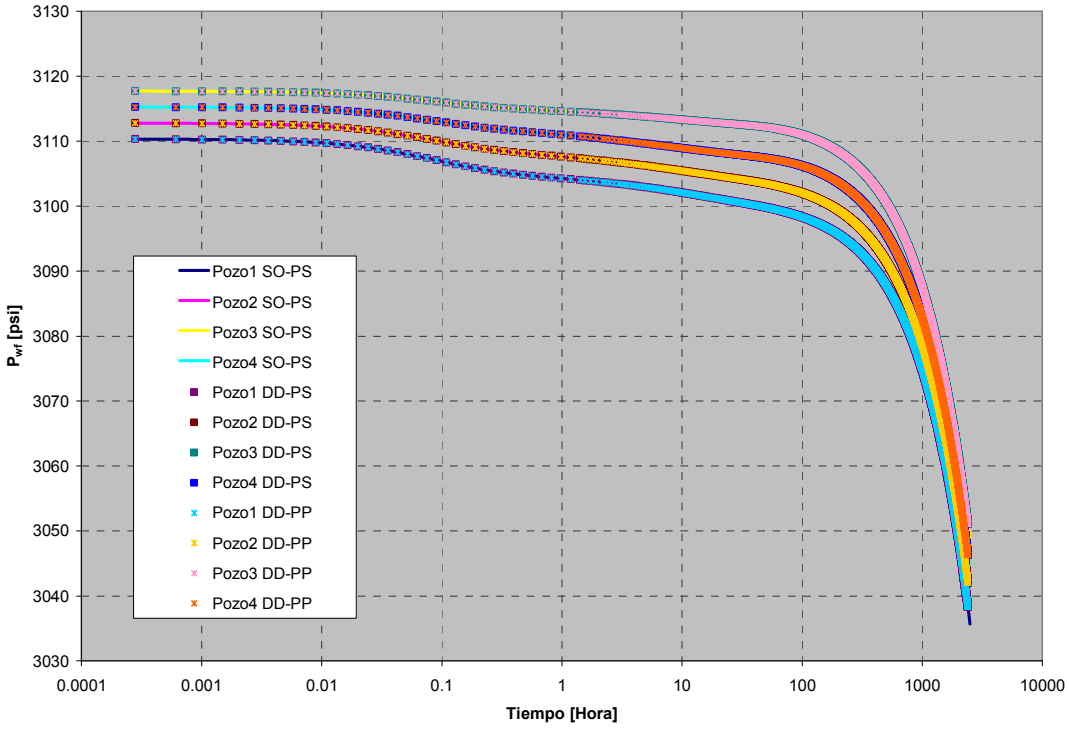


Figura 5.3. Comportamiento de la Pwf contra el tiempo, para los cuatro pozos productores del **caso 2**.

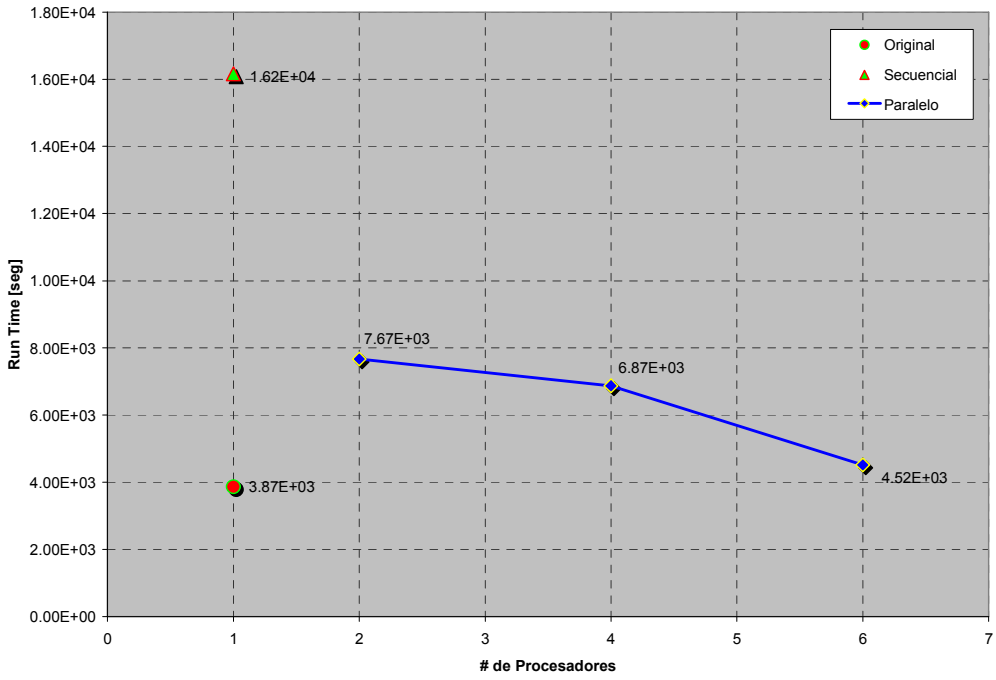


Figura 5.4. Tiempo de corrida (Run Time) registrado para el **caso 2** con los diferentes simuladores.

Tabla 5.3. Datos del yacimiento, fluido y pozo productor empleados en los diferentes simuladores para el **caso 2.**

Datos Generales del problema			
L_X	: 3000.0	μ	: 0.920
L_Y	: 3000.0	API	: 40.0
L_Z	: 500.0	C_f	: 6.9E-06
N_x	: 30	C_r	: 1.0E-06
N_y	: 30	ϕ	: 0.15
N_z	: 5	Bo	: 1.1
N_{WP}	: 4	P_r	: 500.0
N_{WI}	: 1	P_i	: 3000.0
T_{sim}	: 100.0		
K_X	: 100.0		
K_Y	: 100.0		
K_Z	: 100.0		

TABLA 5.4. Datos de los pozos para el **caso 2.**

Pozos Productores

Pozo	X_p	Y_p	Z_p	r_{wp}	Q_{op}
1	3	3	2	0.45	-300.0
2	3	29	2	0.45	-250.0
3	29	3	2	0.45	-150.0
4	29	29	2	0.45	-200.0

Pozos Inyectores

Pozo	X_p	Y_p	Z_p	r_{wp}	Q_{op}
1	16	16	1	0.45	+350.0

5.3) Caso 3.

Simulación de un arreglo de cinco pozos para una malla de 41x41x5:

En este caso se simuló una malla de 41x41x5 nodos, la cual genera un sistema de ecuaciones lineales de 8405 incógnitas, que representa un yacimiento hipotético de 4100x4100x500 [pies^3], con un arreglo de cinco pozos similar al de los dos casos anteriores.
Este caso de simulación se realizó con el fin de mostrar las ventajas que tiene el simulador de Descomposición de Dominio y Programación en Paralelo sobre el simulador original con sistemas (mallas) grandes.

Los datos generales del yacimiento y de los pozos se presentan en las **tablas 5.5 y 5.6**, respectivamente**.**

En este caso se utilizaron sólo seis procesadores para correr el simulador paralelo, y no dos ni cuatro, debido a que es el esquema paralelo que menor tiempo de ejecución tiene.

En la **figura 5.5** se muestra el comportamiento de la P_{wf} contra el tiempo para un solo pozo (pozo 1), obtenido con el simulador de Descomposición de Dominio y Programación en Paralelo y con el simulador original, con el fin de observar el buen ajuste de los resultados.

En la **figura 5.6** se presentan los tiempos de ejecución obtenidos con el simulador original y el simulador de Descomposición de Dominio y Programación en Paralelo para los casos 1, 2 y 3. No se muestran los tiempos obtenidos con el simulador secuencial de Descomposición de Dominio debido a que, como ya se mostró en las **figuras 5.2** y **5.4,** requiere de un mayor tiempo de ejecución.

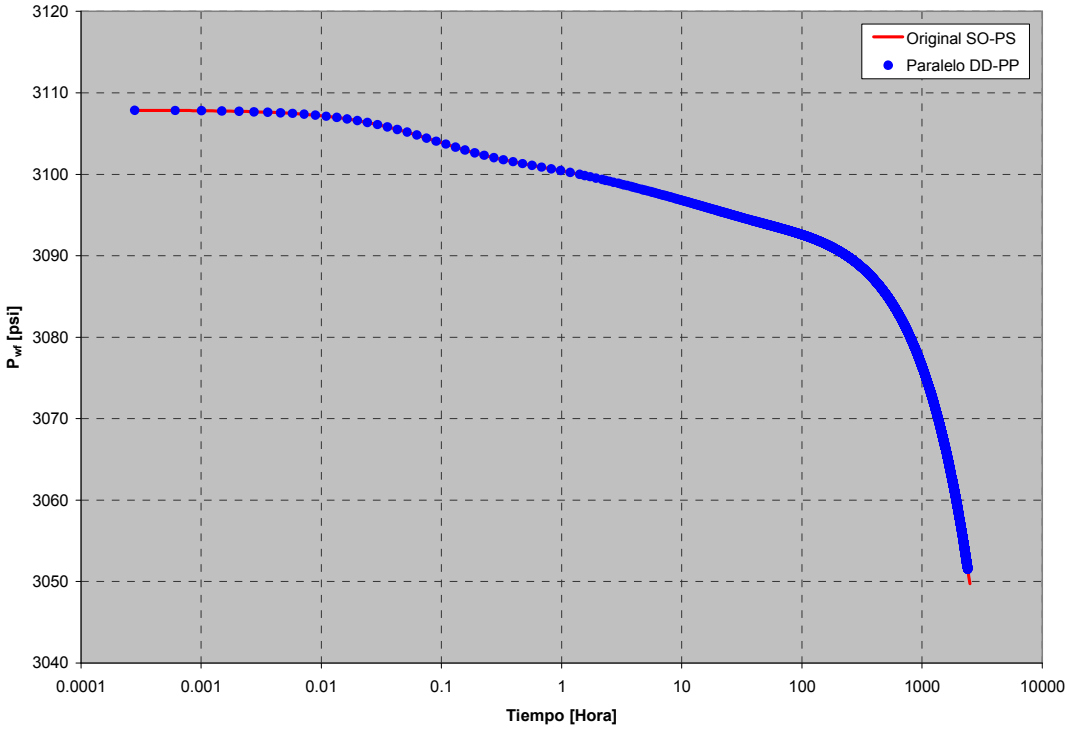


Figura 5.5. Comportamiento de la Pwf contra el tiempo, para el pozo 1 del **caso 3**.

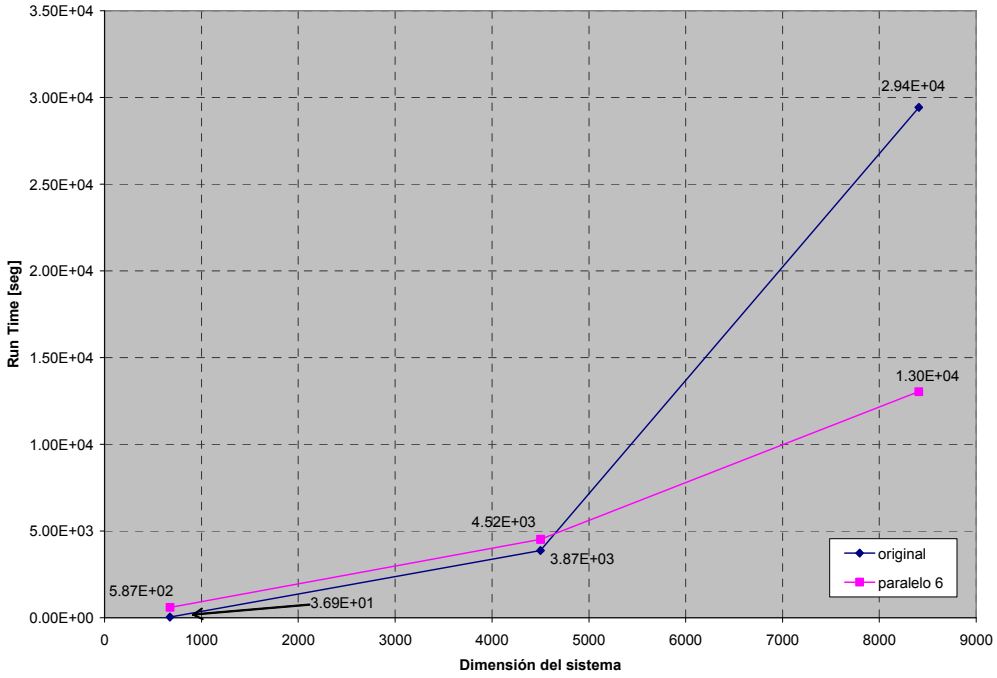


Figura 5.6. Comportamiento del tiempo de corrida contra la dimensión del sistema de ecuaciones global para el simulador de Descomposición de Dominio con Programación en Paralelo y para el simulador original.

Tabla 5.5. Datos del yacimiento, fluido y pozo productor empleados en los diferentes simuladores para el **caso 3.**

Datos Generales del problema			
L_X	: 4100.0	μ	: 0.920
L_Y	: 4100.0	API	: 40.0
L_Z	: 500.0	C_f	: 6.9E-06
N_x	: 41	C_r	: 1.0E-06
N_y	: 41	ϕ	: 0.15
N_z	: 5	Bo	: 1.1
N_{WP}	: 4	P_r	: 500.0
N_{WI}	: 1	P_i	: 3000.0
T_{sim}	: 100.0		
K_X	: 100.0		
K_Y	: 100.0		
K_Z	: 100.0		

TABLA 5.6. Datos de los pozos para el **caso 3.**

Pozos Productores

Pozo	X_p	Y_p	Z_p	r_{wp}	Q_{op}
1	2	2	2	0.45	-350.0
2	2	40	2	0.45	-300.0
3	40	2	2	0.45	-200.0
4	40	40	2	0.45	-250.0

Pozos Inyectores

Pozo	X_p	Y_p	Z_p	r_{wp}	Q_{op}
1	21	21	1	0.45	+400.0

5.4) Discusión de Resultados.

De acuerdo a los casos mostrados se pueden hacer los siguientes comentarios:

- Como se puede apreciar en las **figuras 5.1**, **5.3 y 5.5** la exactitud de los resultados obtenidos con la Descomposición de Dominio es bastante buena, lo cual indica que si se utilizan los parámetros de descomposición adecuados no se presentan las diferencias en la solución discutidas en el **Capítulo IV** y, por lo tanto, los resultados son similares a los del simulador original.

- Las **figuras 5.2** y **5.4** indican que la paralelización optimiza el tiempo de ejecución de los programas. Como se puede ver, el simulador con Descomposición de Dominio se tarda mucho más que el simulador original, pero, una vez paralelizado su rendimiento se optimiza considerablemente, llegando a tener tiempos de corrida muy cercanos a los del simulador original, e incluso menores.

- Se puede ver en la **figura 5.6** que el uso de la Descomposición de Dominio con Programación en Paralelo es recomendable con sistemas grandes. Con sistemas pequeños el simulador original tiene el mejor rendimiento, pues sus tiempos de ejecución son muy pequeños, pero, conforme la dimensión de los sistemas va creciendo, el simulador con Descomposición de Dominio y Programación en Paralelo alcanza un mejor desempeño. Esto se debe en gran medida a que el costo de comunicación por mensajes se reduce conforme el tamaño del sistema aumenta, es decir, mientras más información se mande dentro de un mensaje mayor es el rendimiento mejorado en la ejecución del simulador.

CAPÍTULO VI. CONCLUSIONES Y RECOMENDACIONES.

6.1. CONCLUSIONES

1) La Descomposición de Dominio se puede aplicar satisfactoriamente a la Simulación Numérica de Yacimientos.

2) En la Descomposición sobre el dominio físico se pudo resolver satisfactoriamente para dos subdominios.

3) En la Descomposición sobre el sistema de ecuaciones lineales, para el simulador en que se trabajó y los casos que se analizaron, se observaron los mejores resultados al emplear seis subsistemas en la partición.

4) La Descomposición de Dominio permite resolver sistemas más grandes de lo que se podría en forma directa.

5) La Descomposición de Dominio se puede paralelizar sin alterar la exactitud de sus resultados y optimizando el tiempo de ejecución del simulador.

6) El simulador original (en 1 procesador) es más rápido que el simulador con Descomposición de Dominio y Programación en Paralelo (en 2, 4, y 6 procesadores) para sistemas pequeños. Pero, el simulador con Descomposición de Dominio y Programación en Paralelo (en 6 procesadores) es más rápido que el simulador original para sistemas grandes.

6.2. RECOMENDACIONES

1) Analizar otras alternativas y formulaciones que permitan descomponer el dominio físico en más de dos subdominios.

2) Implementar una subrutina que permita realizar el ordenamiento del menor al mayor índice de la malla de simulación para reducir el ancho de banda del sistema de ecuaciones y así optimizar el tiempo de ejecución.

3) Considerar la información que es ignorada por cada subsistema en el término independiente, con la finalidad de mejorar la exactitud de los resultados y minimizar las iteraciones necesarias para la convergencia reduciendo así el tiempo de ejecución.

4) Considerar la paralelización de otras subrutinas del simulador para reducir aun más el tiempo de corrida.

5) Aplicar el simulador de Descomposición de Dominio y Programación en Paralelo sólo cuando se van a resolver sistemas grandes.

6) Trabajar con un método especializado en la solución de ecuaciones lineales dispersas que sea más rápido que NSPIV para mejorar el tiempo de ejecución, por ejemplo GMRES.

NOMENCLATURA

API	Densidad del aceite.	[°API]
$[A]x = d$	Forma de la ecuación matricial que se soluciona.	---
B	Frontera entre subdominios.	---
bo	Inverso del factor de volumen del aceite.	$\left[\frac{\text{pie}^3}{\text{pie}^3}\right]$
Bo	Factor de Volumen del aceite.	$\left[\frac{\text{pie}^3}{\text{pie}^3}\right]$
bw	Ancho de banda.	
C_f	Compresibilidad de la formación.	[psi^{-1}]
C_r	Compresibilidad de la roca.	[psi^{-1}]
D	Profundidad.	[*L*]
GapE	Distancia de la diagonal principal a la primera diagonal interna.	---
GapN	Distancia de la diagonal principal a la segunda diagonal interna.	---
GapU	Distancia de la diagonal principal a la tercera diagonal interna.	---
In Time	Tiempo promedio de solución de cada subsistema.	[seg]
k	Permeabilidad absoluta.	[mD]
K_X	Permeabilidad en la dirección X.	[mD]
K_Y	Permeabilidad en la dirección Y.	[mD]
K_Z	Permeabilidad en la dirección Z.	[mD]
L_X	Longitud en la dirección X.	[pie]
L_Y	Longitud en la dirección Y.	[pie]
L_Z	Longitud en la dirección Z.	[pie]
N_c	Tamaño de la matriz de coeficientes.	---
NTS	Numero de pasos de tiempo.	---
N_{WI}	Número de pozos inyectores.	---
N_{WP}	Número de pozos productores.	---
N_x	Número de nodos en la dirección X.	---
N_y	Número de nodos en la dirección Y.	---
N_z	Número de nodos en la dirección Z.	---
P_i	Presión inicial del yacimiento.	[psi]
P_r	Presión de referencia.	[psi]
P_{wf}	Presión de fondo fluyendo.	[psi]
Qo	Gasto de producción.	[BPD]
Q'$_o$	Pseudogasto entre los subdominios.	$\left[\frac{L^3}{t}\right]$
$\tilde{q}_s$	Gasto volumétrico @ CS por unidad de volumen de roca.	$\left[\frac{L^3}{tL^3}\right]$
r_{wi}	Radio del pozo inyector.	[pie]
r_{wi}	Radio del pozo productor.	[pie]
Run Time	Tiempo de total corrida.	[seg]
T_{sim}	Tiempo total de simulación.	[días]
Vr	Volumen de roca de la celda.	[L^3]
X_P	Coordenada en X del pozo productor.	---
Y_P	Coordenada en Y del pozo productor.	---
Z_P	Coordenada en Z del pozo productor.	---

Letras griegas

Δt	Incremento de tiempo.	[t]
ϕ	Porosidad.	[Fraccion]
μ	Viscosidad del aceite.	[cp]
Γ	Frontera interna en el traslape.	---
Ω	Dominio sobre el que realiza la descomposción.	---
Ω_i	Subdominio i en la descomposición.	---

Subíndices

i	Dirección en el eje X.
j	Dirección en el eje Y.
k	Dirección en el eje Z.

REFERENCIAS Y LECTURAS RECOMENDADAS.

Aziz, Khalid, "Reservoir Simulation Grids: Opportunities and Problems", SPE 25233, Febrero, 1993, pp. 7-14.

Aziz, Khalid y **Settari,** Antonin, "Petroleum Reservoir Simulation", London: Elsevier Applied Science Publishers, 1979.

Bjørstad, Petter E. y **Kårstad**, Terje, "Domain Decomposition, Parallel Computing and Petroleum Engineering", Domain Based Parallelism and Problem Decomposition Methods in Computational Science and Engineering, pp. 39-56 , Siam, Philadelphia, 1995.

Cabrera F., Eduardo César, **Gordillo R.**, José L. Y **Solís G.**, Omar Alejandro, "Programación de Aplicaciones Numéricas Paralelas", V Semana de Supercómputo, DGSCA – UNAM, México, D.F.

Chan, Tony F. y **Mathew**, Tarek P.**,** "Domain Decomposition Algorithms", Acta Numérica, pp. 61-143, 1994.

Cheng, Wu-Sun**, Jameson,** Anthony **y Mitty,** Todd J.**,** "AIRPLANE on the IBM Parallel System SP1", Parallel Computational Fluid Dynamics; New Algorithms and Applications, N. Satofuca, J. Periaux y A. Ecer, Elsevier, Amsterdam, Holanda, 1995, pp. 205-212.

Craft, B. C. y **Hawkins,** M. F., "Applied Petroleum Reservoir Engineering", Prentice Hall, New Jersey, 1959.

Coats, K. H., "Use and Misuse of Reservoir Simulation Models", JPT, Noviembre 1969, pp. 1391-1398.

Deng, Yuetan y **Glimm**, James, "Fluid Dynamics Using Interface Methods on Parallel Processors", Parallel Computational Fluid Dynamics; Implementations and Results, Horst Simon D., Scientific and Engineering Computation Series, The MIT Press Cambridge, Massachusetts, Londres, Inglaterra, 1992, pp. 257-270.

Ertekin T., **Abou-Kassem** J. H. y **King** G. R., "Basic Applied Reservoir Simulation", SPE, Textbook Series, Vol.7, Richardson, Texas 2001.

Gordillo R., José L., "Taller de Envío de Mensajes (MPI)", Clusters de Alto Rendimiento, DGSCA - UNAM, México, D.F., 21 y 22 de junio de 2001.

Gropp, William, **Lusk,** Ewing y **Skjellum,** Anthony, "Using MPI", The MIT Press Cambridge, Massachusetts, Londres, Inglaterra.

Gropp, William y **Smith**, Barry, "Parallel Domain Decomposition Software", Domain Based Parallelism and Problem Decomposition Methods in Computational Science and Engineering, pp. 97-106 , Siam, Philadelphia, 1995.

Hernández R., O. y **Del Valle M.**, P., Tesis: "Simulador Numérico de Aceite con Interfaz de Visualización para uso Académico", UNAM, México, D.F., Abril 2005.

Jean-Yves, Berthou y **Laurent**, Colombet, "Which Approach to Parallelizing Scientific Codes, That is the Question", Parallel Computing, Elsevier, Francia, 1997.

Kohlberger, Timo**, Schnör,** Christoph**, et.al.,** "Parallel Variational Motion Estimation by Domain Decomposition and Cluster Computing", Springer-Verlag Berlin Heidelberg, pp. 205 216, 2004.

MacDonald, Neil, **Minty,** Elspeth, et. al., "Writing Message Passing Parallel Programs with MPI, A Two Day Course on MPI Usage", Course Notes Version 1.8.2, EPCC, The University of Edinburgh.

Matax C. C. y **Dalton**, R., "Reservoir Simulation", SPE, Monograph, Vol.13, Richardson, Texas 1990.

Odeh, A. S., "Reservoir Simulation… What is it?", JPT, pp. 1383-1388.

Ortega A., Jorge L., Tesis: "Architectural Patterns for Parallel Programming, Models for Performance Estimation", Department of Computer Science, University of London, Londres, Inglaterra, Marzo de 2000.

Ortega A., Jorge L., Tesis: "Architectural Patterns for Parallel Programming", Department of Computer Science, University of London, Londres, Inglaterra, Septiembre de1998.

Ortega A., Jorge L., "The Shared Resource Pattern, An Activity Parallelism Architectural Pattern for Parallel Programming", Departamento de Matemáticas, Facultad de Ciencias, UNAM, México, D.F., 2003.

Pancake, Cherri M., "Is Parallelism for You?", Oregon State University, IEEE Computational Science & Engineering, 1996.

Quarteroni, Alfio y **Valli,** Alberto**,** "Domain Decomposition Methods for Partial Diferential Equations", Oxford University Press, Oxford, Inglaterra, 1999.

Schreiber, Robert y **Simon**, Horst D., "Towards the Teraflops Capability for CFD", Parallel Computational Fluid Dynamics; Implementations and Results, Horst Simon D., Scientific and Engineering Computation Series, The MIT Press Cambridge, Massachusetts, Londres, Inglaterra, 1992, pp. 313-341.

Smith, Barry F.**,** "Domain Decomposition, Parallel Multilevel Methods for Elliptic Partial Differential Equations", Cambridge University Press, 1996. [a]

Smith, Barry F.**,** "Domain Decomposition Methods for Partial Differential Equations", Mathematics and Computer Science Division Argonne National Laboratory. [b]

Staggs, H. M. y **Herbeck**, E. F., "Reservoir Simulation Models – An Engineering Overview", JPT, Diciembre 1971, pp. 1428 1436.

Thomas, G. W., "The Role of Reservoir Simulation in Optimal Reservoir Management", JPT, Marzo 1985, pp. 13-22.

Toselli, Andrea y **Widlund,** Olof, "Domain Decomposition Methods – Algorithms and Theory", Springer Series in Computational Mathematics, Vol. 34, 2005.

Watts, J. W, "Reservoir Simulation: Past, Present and Future", SPE 38441, Junio 1997, pp. 333-342.

Wheeler, Mary F., **Arbogast,** Todd, et. al., "A Parallel Multiblock/Multidomain Approach for Reservoir Simulation", SPE 51884, 1999.

Wohlmuth, Barbara I., "Discretization Methods and Iterative Solvers Based on Domain Decomposition", Springer, Berlin, 2001.

Direcciones electrónicas:

1. www.ddm.org (agosto 2005)

Cualquier duda o sugerencia respecto a este trabajo:

Rodrigo Orantes López rolorant2@yahoo.com.mx

Uriel Cedillo Trejo urielcedillo@gmail.com