Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Tesis de Licenciatura

“Estudio de la paralelización de un sistema de reservorio de petróleo bidimensional de flujo bifásico petróleo-agua”

Tesista: Javier Langone LU 112/95 (jlangone@dc.uba.ar)

Directora: Ing. Marisa Bauzá, M. S. (mbauza@dc.uba.ar)

Resumen

La simulación numérica de reservorios de petróleo es una herramienta importante para entender el comportamiento de un reservorio de petróleo y poder plantear diferentes estrategias de recuperación. Tradicionalmente, la simulación de reservorios grandes y complejos se realizaba en supercomputadoras, pero la aparición de los clusters de PCs ha demostrado ser una alternativa de bajo costo para realizar estas simulaciones utilizando la computación paralela.

En este trabajo se estudian las características de un reservorio bidimensional de flujo bifásico *petróleo-agua*, se analiza la modelización de dicho reservorio y la solución del sistema de ecuaciones del modelo matemático obtenido. Se presentan algunos métodos iterativos para la resolución de dichas ecuaciones. Se plantea una solución secuencial para verificar el correcto funcionamiento del simulador y se implementan varias técnicas de paralelización en algunos de los métodos iterativos estudiados.

El objetivo de este trabajo es analizar la reducción en los tiempos de procesamiento entre una versión serial, una versión paralela que utiliza primitivas de comunicación bloqueante y otra versión paralela que utiliza primitivas de comunicación no bloqueante. Para las pruebas de performance de las versiones paralelizadas del simulador se utiliza un cluster de PC heterogéneo, con comunicación entre procesos vía una red Fast Ethernet, y una máquina de dos procesadores con memoria compartida.

Encontramos que uno de los algoritmos elegidos para resolver el sistema de ecuaciones lineales, el procedimiento fuertemente implícito (SIP), fue el que mejores resultados produjo, tanto a nivel de tiempos de ejecución como de convergencia.

Luego del desarrollo y optimización de las implementaciones paralelas, mediante un refinamiento iterativo, se lograron importantes reducciones en el tiempo de cálculo con respecto a la ejecución secuencial, lográndose en algunos casos una performance muy cercana la ideal.

Abstract

Numerical simulation of petroleum reservoirs is an important tool to understand the behavior of an oil reservoir and its recovery with different strategies. Traditionally, large scale reservoir simulation had been performed on supercomputers. With the advent of cluster computing, these simulations can be performed efficiently and cost effective in clusters of PC using parallel computing.

In this work, we study the properties of a two dimensional, two phase *oil-water* flow system, we modelize this reservoir simulator and we analize a solution to the system of equations of the mathematical model. We present some iterative methods to solve the system of linear equations. We introduce a sequential solution of the reservoir simulator to verify the correct behavior of the solution and we implement different parallelization techniques in some of the iteratives methods that we studied.

The goal of this work is to analize the reduction of the processing time among a serial version, a blocking parallel version and a nonblocking parallel version of the simulator. Performance tests were run on a dedicated Fast Ethernet heterogeneous cluster of PC and a two processor shared memory supercomputer.

We found that one of the chosen iterative methods to solve the system of linear equations, named Strongly Implicit Procedure (SIP), produced the best results: better processing time reduction and fast convergence.

After the development and optimization of the parallel versions, which involved an iterative refinement, we achieved significant time reduction compared to the serial version, achieving in some cases a performance very close to the ideal.

Agradecimientos

Deseo agradecer en primer lugar a mi directora de tesis, Ing. Marisa Bauzá, por su apoyo personal y sus valiosas sugerencias que me guiaron a lo largo de todo el desarrollo de este trabajo de investigación.

A mi familia por su paciencia y comprensión.

A Pablo Milano, por brindarme soporte a todas mis dudas y compartir conmigo sus conocimientos y anécdotas en cuanto a programación paralela se refiere.

A María de los Angeles Morelli por su buena predisposición y por facilitarme todos los artículos y libros que necesité.

A Esteban Mocskos por darme toda la ayuda necesaria para poder trabajar con el cluster Beowulf.

Indice General

Capítulo 1

Introducción

La simulación numérica de los reservorios de petróleo ocupa un rol primordial en la Industria del Petróleo ya que permite optimizar la recuperación de este valioso mineral, hacer estimaciones sobre la calidad de los reservorios y planear estrategias de producción. Las simulaciones de reservorios en máquinas paralelas tienen el potencial de resolver problemas más grandes y realistas que los que se podían resolver en el pasado. Tradicionalmente, la simulación de reservorios grandes y complejos se realizaba en supercomputadoras, pero la aparición de los clusters de PCs ha demostrado ser una alternativa de bajo costo para realizar este tipo de simulaciones.

En este trabajo se presenta el desarrollo de un simulador de reservorio de petróleo bidimensional, en coordenadas cartesianas, de flujo bifásico *petróleo-agua* y posteriormente un estudio sobre la paralelización de dicho simulador. Además del estudio sobre la modelización del reservorio y su implementación, uno de los objetivos planteados aquí es analizar la reducción en los tiempos de procesamiento comparando una versión serial, una versión paralela que utiliza primitivas de comunicación bloqueante y otra versión paralela que utiliza primitivas de comunicación no bloqueante.

El modelo matemático utilizado se basa en combinar las ecuaciones de Darcy [38] para cada fase (oleosa y acuosa), las ecuaciones de continuidad para cada componente y las ecuaciones de estado empíricas. Estas últimas se basan en la medición en el laboratorio de los parámetros PVT (presión-volumen-temperatura) del petróleo. El sistema de ecuaciones resultantes para el flujo bifásico bidimensional se resuelve aplicando la técnica de diferencias finitas con el método IMPES (Implícito en Presiones y Explícito en Saturaciones) [6, 14, 38]. Este método es uno de los más utilizados en los simuladores de reservorios de petróleo de uso comercial. La idea principal consta de dos pasos. El primero combina las ecuaciones diferenciales parciales para lograr una sola ecuación en presión, dicha ecuación se linealiza y se resuelve aplicando un esquema implícito. El segundo paso resuelve las saturaciones explícitamente.

Además, se analizan distintos métodos iterativos para resolver el conjunto de ecuaciones lineales del primer paso del método IMPES. Para ello, se tomaron en cuenta dos métodos comúnmente usados, el BSOR [38] o método de sobrerrelajación en bloques y el SIP [37] o procedimiento fuertemente implícito en sus siglas en inglés.

Cluster de PC, máquinas multiprocesadores y paralelización

Los clusters de PCs se presentan como una alternativa de bajo costo frente a las supercomputadoras en el momento de encarar un proyecto de cómputo en paralelo. Un cluster de PCs está compuesto por un conjunto de computadoras personales interconectadas por una red privada de alta velocidad. Los nodos del cluster son los encargados de ejecutar los correr procesos paralelos. Por lo general son máquinas heterogéneas [29].

La comunicación entre los distintos nodos se realiza a través de una red dedicada mediante el intercambio de mensajes. No existe el concepto de memoria compartida. Cada proceso tiene su propio espacio de memoria y su propia unidad de procesamiento. Cuando un programador utiliza el paradigma de intercambio de mensajes debe decidir cuándo comunicarse con otros procesos, con quién se debe comunicar y qué se debe comunicar.

Por otra parte, al paralelizar un problema como el estudiado en esta tesis, se deben considerar áreas independientes de cálculo dado que los métodos iterativos son altamente secuenciales. Para ello se estudiaron diferentes tipos de particiones de dominio de acuerdo al método elegido para la paralelización.

Se implementó una versión serial del problema que cumple con los objetivos de simular este tipo de reservorio. Para el caso de las versiones en paralelo, dado que el método es intrínsecamente alineal y acoplado, su implementación no fue trivial. Para la paralelización del método SIP, nos basamos en el trabajo de [32] y [37], con partición del dominio utilizando el modelo *red-black*, que fue la que produjo resultados óptimos. Fue notable la reducción de tiempo en el caso de grillas de $50x50$ y de $100x100$ elementos usadas para probar el sistema, comparadas con su versión serial. También fue evidente que los tiempos de comunicación influyen negativamente en el momento de analizar la performance obtenida utilizando el cluster de PC's. Por otra parte, dado que el cluster es heterogéneo, fue necesario implementar un esquema de balance de carga para compensar las diferencias de performance de las distintas máquinas, no siendo ésto necesario en el caso de trabajar con la máquina de memoria compartida.

Capítulo 2

Introducción teórica

Una simulación es la representación de un proceso o fenómeno mediante otro más simple que permite analizar sus características. Un simulador de reservorio de petróleo permite estudiar el comportamiento de los fluidos dentro de un reservorio de petróleo bajo diferentes circunstancias y poder encontrar técnicas óptimas de producción que maximicen la producción de petróleo. En general, los reservorios de petróleo son inaccesibles y estos simuladores se utilizan para obtener un mejor entendimiento del ambiente y poder predecir fenómenos físicos bajo ciertas restricciones naturales. Los simuladores numéricos sirven entonces para analizar las distintas alternativas de explotación de un yacimiento [38].

El objetivo de este capítulo es describir el modelo matemático *black-oil* en dos dimensiones en el que está basado el simulador del reservorio de petróleo bidimensional de flujo bifásico petróleo-agua que se ha implementado.

Un reservorio de petróleo constituye un medio poroso en donde se encuentran hidrocarburos y agua. En los poros de las rocas se pueden encontrar hasta tres tipos de componentes: agua, petróleo y gas. Para simular el flujo bifásico petróleo-agua se adopta el modelo *black-oil* [6, 38]. En este modelo para flujo bifásico se asume que hay dos componentes de fluidos y dos fases. El componente petróleo y la componente agua no se mezclan, por lo tanto, no existe transferencia de masa entre la fase oleosa y la fase acuosa. La fase acuosa es monocomponente, constituida sólo por la componente agua. Además, se asume que el flujo es isotérmico y las fases están en un estado de equilibrio isotérmico.

La Ingeniería de Reservorios utiliza un vocabulario y unas unidades particulares. Para entender qué hace el simulador es necesario definir ciertos términos y propiedades.

Propiedades de la roca

Una propiedad fundamental de un medio poroso es su habilidad para transportar fluidos. Esta habilidad se conoce con el nombre de *permeabilidad* (k) y nos da una medida de la facilidad con la que los fluidos pueden moverse a través de los poros de la roca.

La *porosidad* (ϕ) de la roca mide la fracción de vacíos existentes en la unidad de volumen de roca, es decir, mide la capacidad de la roca para almacenar fluidos. Si bien es adimensional, suele expresarse en porcentaje.

La *compresibilidad* (c_r) de la formación mide el cambio en el volumen de la roca con la variación de la presión de los fluidos contenidos.

La *permeabilidad* (*k*) de la roca mide la facilidad con la que los fluidos pueden moverse a través de los poros de la roca.

Propiedades de los fluidos

La *densidad* (ρ) se define como la masa de fluido por unidad de volumen.

La *viscosidad* (μ) indica la resistencia que hace un fluido para moverse.

La *compresibilidad* (c_f) de un fluido mide el cambio en el volumen del fluido con respecto a la variación de la presión del mismo.

El *factor de volumen* (B_l) de la fase *l* se define como el cociente entre el volumen ocupado por el fluido en condiciones de reservorio y el volumen que ocupa en condiciones estándar de superficie.

$$B = \frac{\text{volumen en condiciones de reservorio}}{\text{volumen en condiciones standard}} \tag{1}$$

La *solubilidad del gas en el petróleo* ($\mathrm{R_{so}}$) es el volumen de gas que se solubiliza en la unidad de volumen de petróleo a una determinada presión. Se mide en volumen de gas en condiciones estándar por unidad de volumen de petróleo en condiciones estándar.

La *solubilidad del gas en el agua* ($\mathrm{R_{sw}}$) es el volumen de gas que se solubiliza en la unidad de volumen de agua a una determinada presión. Se mide en volumen de gas en condiciones estándar por unidad de volumen de agua en condiciones estándar. La solubilidad del gas en el agua es generalmente pequeña a las presiones del reservorio y muchas veces se desprecia.

La *permeabilidad relativa* y la *presión capilar* son propiedades que gobiernan la interacción entre las rocas del reservorio y los fluidos.

La *permeabilidad relativa* (k_r) mide la reducción de la permeabilidad de los poros de la roca debido a los efectos del flujo multifásico, es decir, es el cociente entre la permeabilidad efectiva de la fase *l* y la permeabilidad absoluta. La *permeabilidad absoluta* se define como la permeabilidad de un medio poroso cuando existe una sola fase.

La permeabilidad relativa de la fase *l* está en el rango $0 \le k_{rl} \le 1$. Por ejemplo, en el sistema *petróleo-agua*, k_{ro} y k_{rw} al ser funciones de S_w tienen la forma como se muestra en la fig. 1.

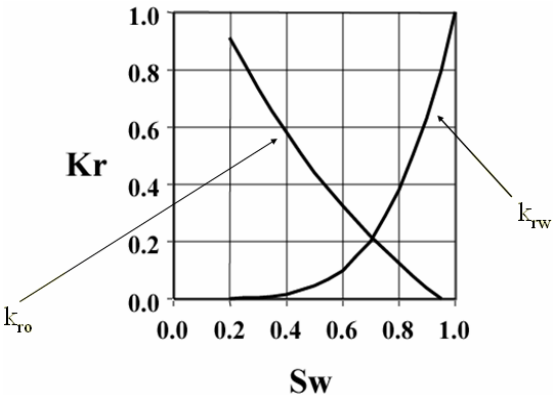

fig. 1. Permeabilidad relativa del sistema *petróleo-agua*.

La *presión capilar* es la diferencia de presión que existe entre dos fluidos que no se mezclan. Para el sistema *petróleo-agua* queda la ecuación planteada como sigue,

$$P_{cwo} = p_o - p_w = f(S_w) \tag{2}$$

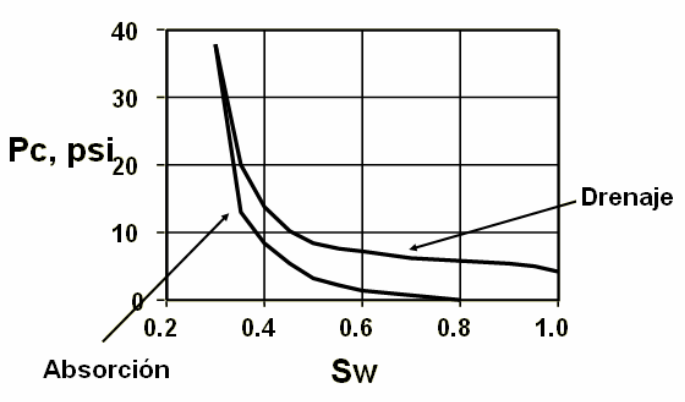

fig. 2. Presión capilar.

Debemos señalar que este trabajo es sobre un estudio de la paralelización y no sobre el petróleo. Por esta razón y para simplificar los cálculos se asume que la presión capilar es nula y se desprecia la compresibilidad.

La *saturación* de una fase se define como la fracción del volumen poral que ocupa esa fase. Como el flujo es bifásico, se tiene la siguiente relación,

$$S_o + S_w = 1 \tag{3}$$

Modelo matemático de flujo en medios porosos

Para el desarrollo del modelo matemático se utilizó la notación de [14].

El modelo matemático de flujo en un reservorio de petróleo está basado en combinar la ecuación de conservación de la masa para cada componente (petróleo y agua) con la ecuación empírica de movimiento de Darcy para cada fase (oleosa y acuosa).

La ecuación de Darcy describe el movimiento de un fluido a través de un medio poroso. Se trata de una ecuación empírica y se define como sigue:

$$\vec{u_p} = -\lambda_p(\nabla P_p + \rho g \nabla D) \tag{4}$$

donde λ_p es el tensor de movilidad de la fase, y $\rho g \nabla D$, el término gravitatorio.

La Ley de Conservación de la Masa dice que la masa no puede ser creada ni destruida.

$$\frac{\partial(\phi \rho_p S_p)}{\partial t} + \nabla(\rho_p \vec{u_p}) + q = 0 \tag{5}$$

donde S_p es la saturación de la fase, q es el caudal másico, $\vec{u}_p$ es la velocidad de la fase y ρ_p es la densidad de la fase.

Las ecuaciones de flujo del petróleo y del agua para un reservorio bifásico en coordenadas cartesianas con inyección de agua surgen de combinar estas ecuaciones.

$$\begin{aligned} &\frac{\partial}{\partial x}\left[\beta_c k_x A_x \frac{k_{rl}}{\mu_l B_l}\left(\frac{\partial p_l}{\partial x} - \gamma_l \frac{\partial Z}{\partial x}\right)\right]\Delta x + \frac{\partial}{\partial y}\left[\beta_c k_y A_y \frac{k_{rl}}{\mu_l B_l}\left(\frac{\partial p_l}{\partial y} - \gamma_l \frac{\partial Z}{\partial y}\right)\right]\Delta y \\ &+ \frac{\partial}{\partial z}\left[\beta_c k_z A_z \frac{k_{rl}}{\mu_l B_l}\left(\frac{\partial p_l}{\partial z} - \gamma_l \frac{\partial Z}{\partial z}\right)\right]\Delta z = \frac{V_b}{\alpha_c}\frac{\partial}{\partial t}\left(\frac{\phi S_l}{B_l}\right) - q_{lsc} \end{aligned} \tag{6}$$

donde $l = o$ o w. (o = oil = petróleo; w = water = agua).

En esta ecuación se desprecia el signo del caudal de extracción de petróleo, q_{osc}, y del caudal de inyección de agua, q_{wsc}.

Como se trabaja con un reservorio horizontal, los gradientes de profundidad para los ejes x e y desaparecen

$$\partial Z \big/ \partial x = \partial Z \big/ \partial y = 0 \tag{7}$$

como así también la coordenada z

$$\frac{\partial}{\partial z}\left[\beta_c k_z A_z \frac{k_{rl}}{\mu_l B_l}\left(\frac{\partial p_l}{\partial z} - \gamma_l \frac{\partial Z}{\partial z}\right)\right]\Delta z = 0 \tag{8}$$

Bajo estos supuestos, la ec. (6) se simplifica:

$$\begin{aligned} &\frac{\partial}{\partial x}\left[\beta_c k_x A_x \frac{k_{rl}}{\mu_l B_l}\left(\frac{\partial p_l}{\partial x}\right)\right]\Delta x + \frac{\partial}{\partial y}\left[\beta_c k_y A_y \frac{k_{rl}}{\mu_l B_l}\left(\frac{\partial p_l}{\partial y}\right)\right]\Delta y \\ &= \frac{V_b}{\alpha_c}\frac{\partial}{\partial t}\left(\frac{\phi S_l}{B_l}\right) - q_{lsc} \end{aligned} \tag{9}$$

Al no haber gas, $S_g = 0$, la relación entre las saturaciones es,

$$S_w + S_o = 1 \tag{10}$$

y al considerar nula la presión capilar del petróleo-agua, se deduce que,

$$P_{cow} = p_o - p_w = f(S_w) = 0 \tag{11}$$

Estas ecuaciones contienen cuatro incógnitas: p_o, p_w, S_w y S_o. A partir de las ecuaciones (10) y (11) se pueden eliminar dos incógnitas:

$$S_o = 1 - S_w \tag{12}$$

y

$$p_o = p_w \tag{13}$$

Sustituyendo estos resultados en la ec. (9), se obtienen las siguientes ecuaciones:

- Para el componente petróleo

$$\frac{\partial}{\partial x}\left[\beta_c k_x A_x \frac{k_{ro}}{\mu_o B_o}\left(\frac{\partial p_o}{\partial x}\right)\right]\Delta x + \frac{\partial}{\partial y}\left[\beta_c k_y A_y \frac{k_{ro}}{\mu_o B_o}\left(\frac{\partial p_o}{\partial y}\right)\right]\Delta y = \frac{V_b}{\alpha_c}\frac{\partial}{\partial t}\left(\frac{\phi(1-S_w)}{B_o}\right) - q_{osc} \tag{14}$$

- Para la componente agua

$$\frac{\partial}{\partial x}\left[\beta_c k_x A_x \frac{k_{rw}}{\mu_w B_w}\left(\frac{\partial p_o}{\partial x}\right)\right]\Delta x + \frac{\partial}{\partial y}\left[\beta_c k_y A_y \frac{k_{rw}}{\mu_w B_w}\left(\frac{\partial p_o}{\partial y}\right)\right]\Delta y = \frac{V_b}{\alpha_c}\frac{\partial}{\partial t}\left(\frac{\phi S_w}{B_w}\right) - q_{wsc} \tag{15}$$

Las ecuaciones (14) y (15) son ecuaciones diferenciales parciales no lineales cuya solución analítica es muy complicada o casi imposible de obtener. Por lo tanto, es necesario aplicar métodos numéricos para resolver el problema de flujo. Se recurre entonces a la aproximación por el método de diferencias finitas. Las ecuaciones de flujo se discretizan mediante aproximaciones algebraicas con las derivadas de segundo orden con respecto al espacio y con las derivadas de primer orden con respecto al tiempo.

Aproximación de las ecuaciones de flujo por el método de diferencias finitas

Discretizar es el proceso de aproximar una ecuación diferencial mediante ecuaciones finitas. Por lo que se reemplazan las derivadas con aproximaciones algebraicas.

Sea *f* una función de dos variables,

$$f = f(x,t) \tag{16}$$

Por definición,

$$\frac{\partial f}{\partial x}(x,t) = \lim_{x \to 0} \frac{f(x+\Delta x,t) - f(x,t)}{\Delta x} \tag{17}$$

Esta derivada se puede aproximar por diferencias finitas “centradas”,

$$\frac{\partial f}{\partial x}(x,t) \cong \frac{f(x+\Delta x,t) - f(x,t)}{\Delta x} \tag{18}$$

El reservorio se modela usando una grilla equiespaciada en el plano *x*-*y* de bloques centrados como se muestra en la fig. 3.

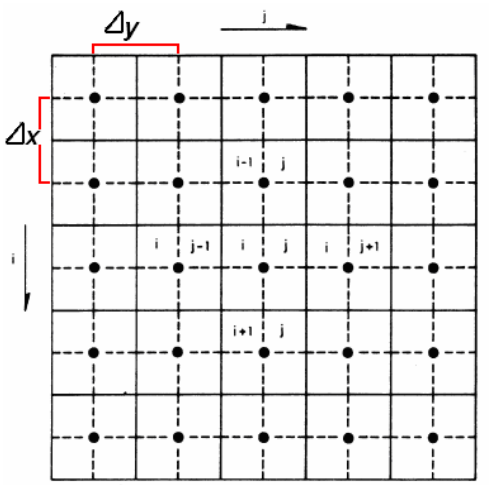

fig. 3. Grilla equiespaciada de bloques centrados

Si, por ejemplo, se aplica diferencias finitas centradas en el punto (*i*, *j*) a la coordenada *x* en el lado izquierdo de las ecuaciones (14) y (15) para aproximar las derivadas de segundo orden mediante una discretización en el espacio, se obtiene:

$$\frac{\partial}{\partial x}\left(\beta_c k_x A_x \frac{k_{rl}}{\mu_l B_l}\left(\frac{\partial p_l}{\partial x}\right)\right)_{i,j} \Delta x_{i,j} = \tag{19}$$

$$\frac{\left(\beta_c k_x A_x \frac{k_{rl}}{\mu_l B_l}\left(\frac{\partial p_l}{\partial x}\right)\right)_{i+\frac{1}{2},j} - \left(\beta_c k_x A_x \frac{k_{rl}}{\mu_l B_l}\left(\frac{\partial p_l}{\partial x}\right)\right)_{i-\frac{1}{2},j}}{\Delta x_{i,j}} \Delta x_{i,j} =$$

$$\frac{\left(\beta_c k_x A_x \frac{k_{rl}}{\mu_l B_l}\left(\frac{p_{o_{i+1,j}} - p_{o_{i,j}}}{\Delta x_{i+\frac{1}{2},j}}\right)\right)_{i+\frac{1}{2},j} - \left(\beta_c k_x A_x \frac{k_{rl}}{\mu_l B_l}\left(\frac{p_{o_{i,j}} - p_{o_{i-1,j}}}{\Delta x_{i-\frac{1}{2},j}}\right)\right)_{i-\frac{1}{2},j}}{\Delta x_{i,j}} \Delta x_{i,j} =$$

$$\left(\beta_c k_x A_x \frac{k_{rl}}{\mu_l B_l}\left(\frac{p_{o_{i+1,j}} - p_{o_{i,j}}}{\Delta x_{i+\frac{1}{2},j}}\right)\right)_{i+\frac{1}{2},j} - \left(\beta_c k_x A_x \frac{k_{rl}}{\mu_l B_l}\left(\frac{p_{o_{i,j}} - p_{o_{i-1,j}}}{\Delta x_{i-\frac{1}{2},j}}\right)\right)_{i-\frac{1}{2},j}$$

donde $l = o$ o w.

Reordenando los términos de la ec. (19)

$$\frac{\partial}{\partial x}\left(\beta_c k_x A_x \frac{k_{rl}}{\mu_l B_l}\left(\frac{\partial p_l}{\partial x}\right)\right)_{i,j} \Delta x_{i,j} = \tag{20}$$

$$\left(\frac{\beta_c k_x A_x \frac{k_{rl}}{\mu_l B_l}}{\Delta x}\right)_{i+\frac{1}{2},j} \left(p_{o_{i+1,j}} - p_{o_{i,j}}\right) - \left(\frac{\beta_c k_x A_x \frac{k_{rl}}{\mu_l B_l}}{\Delta x}\right)_{i-\frac{1}{2},j} \left(p_{o_{i,j}} - p_{o_{i-1,j}}\right)$$

Para la coordenada y se hace el mismo procedimiento para llegar a:

$$\frac{\partial}{\partial y}\left(\beta_c k_y A_y \frac{k_{rl}}{\mu_l B_l}\left(\frac{\partial p_l}{\partial y}\right)\right)_{i,j} \Delta y_{i,j} = \tag{21}$$

$$\left(\frac{\beta_c k_y A_y \frac{k_{rl}}{\mu_l B_l}}{\Delta y}\right)_{i,j+\frac{1}{2}} \left(p_{o_{i,j+1}} - p_{o_{i,j}}\right) - \left(\frac{\beta_c k_y A_y \frac{k_{rl}}{\mu_l B_l}}{\Delta y}\right)_{i,j-\frac{1}{2}} \left(p_{o_{i,j}} - p_{o_{i,j-1}}\right)$$

Para simplificar la notación, se define como transmisibilidad direccional de la fase $l = o$ o w

$$T_{lx} = \beta_c \frac{k_x A_x}{\Delta x} \frac{k_{rl}}{\mu_l B_l} \tag{22}$$

$$T_{ly} = \beta_c \frac{k_y A_y}{\Delta y} \frac{k_{rl}}{\mu_l B_l} \tag{23}$$

Utilizando esta notación en las ecuaciones (20) y (21) se obtiene:

$$\left(\frac{\beta_c k_x A_x \frac{k_{rl}}{\mu_l B_l}}{\Delta x} \right)_{i+\frac{1}{2},j} \left(p_{o_{i+1,j}} - p_{o_{i,j}} \right) - \left(\frac{\beta_c k_x A_x \frac{k_{rl}}{\mu_l B_l}}{\Delta x} \right)_{i-\frac{1}{2},j} \left(p_{o_{i,j}} - p_{o_{i-1,j}} \right) = \tag{24}$$

$$T_{lx_{i+\frac{1}{2},j}} \left(p_{o_{i+1,j}} - p_{o_{i,j}} \right) - T_{lx_{i-\frac{1}{2},j}} \left(p_{o_{i,j}} - p_{o_{i-1,j}} \right) =$$

$$T_{lx_{i-\frac{1}{2},j}} p_{o_{i-1,j}} - \left(T_{lx_{i-\frac{1}{2},j}} + T_{lx_{i+\frac{1}{2},j}} \right) p_{o_{i,j}} + T_{lx_{i+\frac{1}{2},j}} p_{o_{i+1,j}}$$

y

$$\left(\frac{\beta_c k_y A_y \frac{k_{rl}}{\mu_l B_l}}{\Delta y} \right)_{i,j+\frac{1}{2}} \left(p_{o_{i,j+1}} - p_{o_{i,j}} \right) - \left(\frac{\beta_c k_y A_y \frac{k_{rl}}{\mu_l B_l}}{\Delta y} \right)_{i,j-\frac{1}{2}} \left(p_{o_{i,j}} - p_{o_{i,j-1}} \right) = \tag{25}$$

$$T_{ly_{i,j+\frac{1}{2}}} \left(p_{o_{i,j+1}} - p_{o_{i,j}} \right) - T_{ly_{i,j-\frac{1}{2}}} \left(p_{o_{i,j}} - p_{o_{i,j-1}} \right) =$$

$$T_{ly_{i,j-\frac{1}{2}}} p_{o_{i,j-1}} - \left(T_{ly_{i,j-\frac{1}{2}}} + T_{ly_{i,j+\frac{1}{2}}} \right) p_{o_{i,j}} + T_{ly_{i,j+\frac{1}{2}}} p_{o_{i,j+1}}$$

Combinando estos resultados, el lado izquierdo de las ecuaciones (14) y (15) se aproxima mediante diferencias finitas centradas en el punto (*i*, *j*) de la siguiente manera:

$$\frac{\partial}{\partial x}\left(\beta_c k_x A_x \frac{k_{rl}}{\mu_l B_l} \left(\frac{\partial p_o}{\partial x} \right) \right)_{i,j} \Delta x_{i,j} + \frac{\partial}{\partial y}\left(\beta_c k_y A_y \frac{k_{rl}}{\mu_l B_l} \left(\frac{\partial p_o}{\partial y} \right) \right)_{i,j} \Delta y_{i,j} = \tag{26}$$

$$T_{lx_{i-\frac{1}{2},j}} p_{o_{i-1,j}} - \left(T_{lx_{i-\frac{1}{2},j}} + T_{lx_{i+\frac{1}{2},j}} + T_{ly_{i,j-\frac{1}{2}}} + T_{ly_{i,j+\frac{1}{2}}} \right) p_{o_{i,j}} + T_{lx_{i+\frac{1}{2},j}} p_{o_{i+1,j}} + T_{ly_{i,j-\frac{1}{2}}} p_{o_{i,j-1}} + T_{ly_{i,j+\frac{1}{2}}} p_{o_{i,j+1}}$$

El lado derecho de las ecuaciones (14) y (15) son términos con derivadas de primer orden en el tiempo y se aproximan mediante una discretización en el tiempo. Es necesario, dividir la coordenada temporal en pasos de tiempo discretos.

En general, la aproximación por diferencias finitas de la derivada de primer orden para la coordenada temporal se calcula de la siguiente manera:

$$\frac{\partial}{\partial t}(f) \approx \frac{1}{\Delta t}\Delta_t(f) \equiv \frac{1}{\Delta t}\left(f^{n+1} - f^{n}\right) \tag{27}$$

La función *f* en general puede ser expresada como

$$f = UVY \tag{28}$$

donde $U \equiv \phi$, $V \equiv 1/B_l$ y $Y \equiv S_l$, donde *l* = *o* o *w*.

Sustituyendo la ec. (28) en

$$\Delta_t f = f^{n+1} - f^{n} \tag{29}$$

se obtiene la diferencia temporal de una función

$$\Delta_t\left(UVY\right) = \left(UVY\right)^{n+1} - \left(UVY\right)^{n} \tag{30}$$

de la que se deriva la siguiente fórmula de expansión conservadora

$$\Delta_t\left(UVY\right) = \left(VY\right)^{n}\Delta_t U + U^{n+1}Y^{n}\Delta_t V + \left(UV\right)^{n+1}\Delta_t Y \tag{31}$$

Como se está trabajando con ecuaciones diferenciales fuertemente no lineales, es necesario hacer esta expansión conservadora para preservar la masa.

Sustituyendo las definiciones de *U, V* e *Y* en la ec. (31) se obtiene,

- Para la saturación de petróleo

$$\Delta_t\left[\phi\left(\frac{1}{B_o}\right)\left(S_o\right)\right] = \left(\frac{S_o}{B_o}\right)^{n}\Delta_t\phi + \phi^{n+1}\left(S_o^{n}\right)\Delta_t\left(\frac{1}{B_o}\right) + \left(\frac{\phi}{B_o}\right)^{n+1}\Delta_t\left(S_o\right) \tag{32}$$

como $S_o = 1 - S_w$ y $\Delta_t S_o = -\Delta_t S_w$, entonces

$$\Delta_t\left[\phi\left(\frac{1}{B_o}\right)\left(1-S_w\right)\right] = \left(\frac{1-S_w}{B_o}\right)^{n}\Delta_t\phi + \phi^{n+1}\left(1-S_w^{n}\right)\Delta_t\left(\frac{1}{B_o}\right) - \left(\frac{\phi}{B_o}\right)^{n+1}\Delta_t\left(S_w\right) \tag{33}$$

- Para la saturación de agua

$$\Delta_t\left[\phi\left(\frac{1}{B_w}\right)\left(S_w\right)\right] = \left(\frac{S_w}{B_w}\right)^{n}\Delta_t\phi + \phi^{n+1}\left(S_w^{n}\right)\Delta_t\left(\frac{1}{B_w}\right) + \left(\frac{\phi}{B_w}\right)^{n+1}\Delta_t\left(S_w\right) \tag{34}$$

Bajo el supuesto de que ϕ es constante,

$$\Delta_t\phi = 0 \tag{35}$$

La diferencia temporal de la función $\left(1/B_l\right)$, donde *l* = *o* o *w*, puede ser expresada de la siguiente manera:

$$\Delta_t\left(\frac{1}{B_l}\right) = \left(\frac{1}{B_l}\right)'\Delta_t p_o \tag{36}$$

donde

$$\left(\frac{1}{B_l}\right)' = \frac{\left(1/B_l^{n+1}\right)-\left(1/B_l^{n}\right)}{\left(p_o^{n+1}-p_o^{n}\right)} \tag{37}$$

Reescribiendo las ecuaciones (32) y (34) con estos resultados se obtiene:

$$\Delta_t\left[\phi\left(\frac{1}{B_o}\right)\left(1-S_w\right)\right] = \phi^{n+1}\left(1-S_w^n\right)\left(\frac{1}{B_o}\right)'\Delta_t p_o - \left(\frac{\phi}{B_o}\right)^{n+1}\Delta_t\left(S_w\right) \tag{38}$$

y

$$\Delta_t\left[\phi\left(\frac{1}{B_w}\right)\left(S_w\right)\right] = \phi^{n+1}\left(S_w^n\right)\left(\frac{1}{B_w}\right)'\Delta_t p_o + \left(\frac{\phi}{B_w}\right)^{n+1}\Delta_t\left(S_w\right) \tag{39}$$

Para el lado derecho de la ec. (14), la aproximación por diferencias finitas para la coordenada temporal en el punto (*i, j*) queda:

$$\frac{V_{b_{i,j}}}{\alpha_c}\frac{\partial}{\partial t}\left(\frac{\phi\left(1-S_w\right)}{B_o}\right)_{i,j} - q_{osc_{i,j}} = \tag{40}$$

$$\frac{V_{b_{i,j}}}{\alpha_c}\frac{1}{\Delta t}\Delta_t\left(\frac{\phi\left(1-S_w\right)}{B_o}\right)_{i,j} - q_{osc_{i,j}} \equiv \frac{V_{b_{i,j}}}{\alpha_c\Delta t}\left(\left(\frac{\phi\left(1-S_w\right)}{B_o}\right)_{i,j}^{n+1} - \left(\frac{\phi\left(1-S_w\right)}{B_o}\right)_{i,j}^{n}\right) - q_{osc_{i,j}} =$$

$$\frac{V_{b_{i,j}}}{\alpha_c\Delta t}\left(\phi^{n+1}\left(1-S_w^n\right)\left(\frac{1}{B_o}\right)'\Delta_t p_o - \left(\frac{\phi}{B_o}\right)^{n+1}\Delta_t\left(S_w\right)\right)_{i,j} - q_{osc_{i,j}}$$

Para el lado derecho de la ec. (15), la aproximación por diferencias finitas para la coordenada temporal queda como sigue:

$$\frac{V_{b_{i,j}}}{\alpha_c}\frac{\partial}{\partial t}\left(\frac{\phi S_w}{B_w}\right)_{i,j} - q_{wsc_{i,j}} \approx \tag{41}$$

$$\frac{V_{b_{i,j}}}{\alpha_c}\frac{1}{\Delta t}\Delta_t\left(\frac{\phi S_w}{B_w}\right)_{i,j} - q_{wsc_{i,j}} \equiv \frac{V_{b_{i,j}}}{\alpha_c\Delta t}\left(\left(\frac{\phi S_w}{B_w}\right)_{i,j}^{n+1} - \left(\frac{\phi S_w}{B_w}\right)_{i,j}^{n}\right) - q_{wsc_{i,j}} =$$

$$\frac{V_{b_{i,j}}}{\alpha_c\Delta t}\left(\phi^{n+1}\left(S_w^n\right)\left(\frac{1}{B_w}\right)'\Delta_t p_o + \left(\frac{\phi}{B_w}\right)^{n+1}\Delta_t\left(S_w\right)\right)_{i,j} - q_{wsc_{i,j}}$$

Cuando estas ecuaciones se escriben para todos los puntos de la grilla del reservorio, $i = 1, 2, ..., N_x$ y $j = 1, 2, ..., N_y$, y son evaluadas en el paso de tiempo *n*+1 (diferencias

atrasadas), representan la aproximación por diferencias finitas de las ecuaciones (14) y (15) para todo el reservorio.

- Para el componente petróleo

$$T^{n+1}_{ox_{i-1/2,j}}\, p^{n+1}_{o_{i-1,j}} - \left(T^{n+1}_{ox_{i-1/2,j}} + T^{n+1}_{ox_{i+1/2,j}} + T^{n+1}_{oy_{i,j-1/2}} + T^{n+1}_{oy_{i,j+1/2}}\right) p^{n+1}_{o_{i,j}} + T^{n+1}_{ox_{i+1/2,j}}\, p^{n+1}_{o_{i+1,j}} + T^{n+1}_{oy_{i,j-1/2}}\, p^{n+1}_{o_{i,j-1}} + T^{n+1}_{oy_{i,j+1/2}}\, p^{n+1}_{o_{i,j+1}} = \frac{V_{b_{i,j}}}{\alpha_c \Delta t}\left(\phi^{n+1}\left(1-S^n_w\right)\left(\frac{1}{B_o}\right)' \Delta_t p_o - \left(\frac{\phi}{B_o}\right)^{n+1} \Delta_t\left(S_w\right)\right)_{i,j} - q^{n+1}_{osc_{i,j}} \tag{42}$$

- Para la componente agua

$$T^{n+1}_{wx_{i-1/2,j}}\, p^{n+1}_{o_{i-1,j}} - \left(T^{n+1}_{wx_{i-1/2,j}} + T^{n+1}_{wx_{i+1/2,j}} + T^{n+1}_{wy_{i,j-1/2}} + T^{n+1}_{wy_{i,j+1/2}}\right) p^{n+1}_{o_{i,j}} + T^{n+1}_{wx_{i+1/2,j}}\, p^{n+1}_{o_{i+1,j}} + T^{n+1}_{wy_{i,j-1/2}}\, p^{n+1}_{o_{i,j-1}} + T^{n+1}_{wy_{i,j+1/2}}\, p^{n+1}_{o_{i,j+1}} = \frac{V_{b_{i,j}}}{\alpha_c \Delta t}\left(\phi^{n+1}\left(S^n_w\right)\left(\frac{1}{B_w}\right)' \Delta_t p_o + \left(\frac{\phi}{B_w}\right)^{n+1} \Delta_t\left(S_w\right)\right)_{i,j} - q^{n+1}_{wsc_{i,j}} \tag{43}$$

Definiendo los coeficientes

$$C_{op} = \frac{V_b}{\alpha_c \Delta t}\left(\phi^{n+1}\left(\frac{1}{B_o}\right)'\right)\left(1-S^n_w\right) \tag{44}$$

$$C_{ow} = -\frac{V_b}{\alpha_c \Delta t}\left(\frac{\phi}{B_o}\right)^{n+1} \tag{45}$$

$$C_{wp} = \frac{V_b}{\alpha_c \Delta t}\left(\phi^{n+1}\left(\frac{1}{B_w}\right)'\right) S^n_w \tag{46}$$

$$C_{ww} = \frac{V_b}{\alpha_c \Delta t}\left(\frac{\phi}{B_w}\right)^{n+1} \tag{47}$$

las ecuaciones (42) y (43) se reescriben en notación compacta para obtener:

- Para el componente petróleo

$$T_{ox_{i-1/2,j}}^{n+1} p_{o_{i-1,j}}^{n+1} - \left(T_{ox_{i-1/2,j}}^{n+1} + T_{ox_{i+1/2,j}}^{n+1} + T_{oy_{i,j-1/2}}^{n+1} + T_{oy_{i,j+1/2}}^{n+1} \right) p_{o_{i,j}}^{n+1} + T_{ox_{i+1/2,j}}^{n+1} p_{o_{i+1,j}}^{n+1} + T_{oy_{i,j-1/2}}^{n+1} p_{o_{i,j-1}}^{n+1} + T_{oy_{i,j+1/2}}^{n+1} p_{o_{i,j+1}}^{n+1} = C_{op_{i,j}} \left(p_{o_{i,j}}^{n+1} - p_{o_{i,j}}^{n} \right) + C_{ow_{i,j}} \left(S_{w_{i,j}}^{n+1} - S_{w_{i,j}}^{n} \right) - q_{osc_{i,j}}^{n+1} \quad (48)$$

- Para la componente agua

$$T_{wx_{i-1/2,j}}^{n+1} p_{o_{i-1,j}}^{n+1} - \left(T_{wx_{i-1/2,j}}^{n+1} + T_{wx_{i+1/2,j}}^{n+1} + T_{wy_{i,j-1/2}}^{n+1} + T_{wy_{i,j+1/2}}^{n+1} \right) p_{o_{i,j}}^{n+1} + T_{wx_{i+1/2,j}}^{n+1} p_{o_{i+1,j}}^{n+1} + T_{wy_{i,j-1/2}}^{n+1} p_{o_{i,j-1}}^{n+1} + T_{wy_{i,j+1/2}}^{n+1} p_{o_{i,j+1}}^{n+1} = C_{wp_{i,j}} \left(p_{o_{i,j}}^{n+1} - p_{o_{i,j}}^{n} \right) + C_{ww_{i,j}} \left(S_{w_{i,j}}^{n+1} - S_{w_{i,j}}^{n} \right) - q_{wsc_{i,j}}^{n+1} \quad (49)$$

Estas ecuaciones se escriben para cada punto (*i*, *j*) del reservorio.

Tratamiento de las transmisibilidades

La transmisibilidad de la fase *l*, *l* = *o* o *w*, del bloque (*i*, *j*) de la grilla con respecto a sus vecinos se define como

$$T_{lx_{i\pm 1/2,j}} = \beta_c \left(\frac{k_x A_x}{\Delta x} \right)_{i\pm 1/2,j} \left(\frac{1}{\mu_l B_l} \right)_{i\pm 1/2,j} k_{rl_{i\pm 1/2,j}} \quad (50)$$

y

$$T_{ly_{i,j\pm 1/2}} = \beta_c \left(\frac{k_y A_y}{\Delta y} \right)_{i,j\pm 1/2} \left(\frac{1}{\mu_l B_l} \right)_{i,j\pm 1/2} k_{rl_{i,j\pm 1/2}} \quad (51)$$

Los signos “+” y “-” en los subíndices se utilizan para identificar los contornos de los bloques de la grilla en las direcciones positiva y negativa de los ejes de coordenadas cartesianas.

La viscosidad, μ, y el factor de volumen, B, dependen de la presión

$$\mu_l = f(p_l) \quad (52)$$

y

$$B_l = f(p_l) \quad (53)$$

Una de las cosas que hace interesante el estudio de reservorios de petróleo es que las propiedades de los fluidos (densidad, viscosidad, etc.) dependen fuertemente de la presión y la temperatura. Las propiedades dependientes de la presión representan alinealidades débiles. Los efectos de estas alinealidades débiles en la estabilidad del modelo dependen de la magnitud de la variación de la presión en un paso de tiempo. La linealización de las

alinealidades débiles generalmente no es crucial para la estabilidad de la solución de las ecuaciones de diferencias finitas [14].

La permeabilidad relativa, k_{rl} , depende de la saturación de la fase *l*.

$$k_{rl} = f(S_l) \tag{54}$$

Las propiedades dependientes de la saturación representan alinealidades fuertes. La linealización de las alinealidades fuertes generalmente es crucial para la estabilidad de la solución de las ecuaciones de diferencias finitas [14].

El método de la interpolación con los parámetros PVT puede utilizarse para calcular en forma explícita tanto la linealización de las alinealidades débiles como la de las fuertes.

Definición 1: Interpolar es el proceso por el que a una tabla de valores se le asocia una expresión matemática que la representa. La función obtenida debe representar de forma exacta los valores de la tabla, pero no proporciona más que una estimación de los valores que no aparezcan en la tabla. La función utilizada es el polinomio de interpolación Newton-Lagrange.

$$f(x) = f(x_0) + \left(f(x_1) - f(x_0)\right)\frac{(x - x_o)}{x_1 - x_o} \tag{55}$$

Se define como movilidad de la fase *l* a:

$$\lambda_{l_{i,j}} = \left(\frac{k_{rl}}{\mu_l B_l}\right)_{i,j} \tag{56}$$

La movilidad queda definida en función de la saturación y también de la presión. La movilidad se ve muy influenciada por la fuerte dependencia con la saturación. Como la grilla quedó discretizada en bloques, los resultados de la simulación van a depender de la manera en que se aproxime este término. La aproximación que se utiliza comúnmente es la denominada *upstream*, o aguas arriba, en el sentido del flujo.

En el reservorio que se está modelando, los términos A_x , A_y , k_x , k_y , Δ_x y Δ_y son constantes, entonces las transmisibilidades se simplifican a:

$$T_{lx_{i\pm 1/2,j}} = \beta_c \left(\frac{k_x A_x}{\Delta x}\right)_{i\pm 1/2,j} \left(\frac{k_{rl}}{\mu_l B_l}\right)_{i\pm 1/2,j} = \beta_c \frac{k_x A_x}{\Delta x}\left(\frac{k_{rl}}{\mu_l B_l}\right)_{i\pm 1/2,j} \tag{57}$$

y

$$T_{ly_{i,j\pm\frac{1}{2}}} = \beta_c \left(\frac{k_y A_y}{\Delta y} \right)_{i,j\pm\frac{1}{2}} \left(\frac{k_{rl}}{\mu_l B_l} \right)_{i,j\pm\frac{1}{2}} = \beta_c \frac{k_y A_y}{\Delta y} \left(\frac{k_{rl}}{\mu_l B_l} \right)_{i,j\pm\frac{1}{2}} \tag{58}$$

Tratamiento de los términos de producción e inyección

En este modelo de reservorio se asume que la tasa de producción de petróleo e inyección de agua son constantes durante toda la explotación del reservorio.

Para los puntos (*i*, *j*) de la grilla en donde no hay un pozo productor, el caudal de extracción es nulo,

$$q_{osc_{i,j}} = 0 \tag{59}$$

Para los puntos donde no hay un pozo inyector, el caudal de inyección es nulo,

$$q_{wsc_{i,j}} = 0 \tag{60}$$

Para un pozo productor de petróleo ubicado en la celda (*i*, *j*), el caudal de extracción es positivo, $q_{osc_{i,j}} > 0$.

Para un pozo inyector de agua ubicado en la celda (*i*, *j*), el caudal de inyección es negativo, $q_{wsc_{i,j}} < 0$.

Condiciones iniciales y de contorno

Las condiciones iniciales establecen presiones y saturaciones constantes al comienzo de la explotación:

$$\left(p_{o_{i,j}} \right)_{t=0} = p_{oINI} \qquad \forall i, j \in \Re^2 \tag{61}$$

$$\left(S_{w_{i,j}} \right)_{t=0} = S_{wINI} \qquad \forall i, j \in \Re^2 \tag{62}$$

$$\left(S_{o_{i,j}} \right)_{t=0} = S_{oINI} \qquad \forall i, j \in \Re^2 \tag{63}$$

Las condiciones de contorno establecen que no hay flujo en los contornos del reservorio, es decir, la transmisibilidad direccional en los bordes del reservorio es nula.

$$T_{x_{1/2,j}} = 0 \qquad j = 1, 2, ..., N_y \tag{64}$$

$$T_{x_{Nx+1/2,j}} = 0 \qquad j = 1, 2, ..., N_y \tag{65}$$

$$T_{y_{i,1/2}} = 0 \qquad i = 1, 2, ..., N_x \tag{66}$$

$$T_{y_{i,N_y+1/2}} = 0 \qquad i = 1, 2, ..., N_x \tag{67}$$

El sistema de ecuaciones diferenciales (14) y (15), con condiciones iniciales (61), (62) y (63); y condiciones de contorno (64), (65), (66) y (67); se resuelve numéricamente mediante la aproximación por diferencias finitas y aplicando el método IMPES.

Análisis de la estabilidad

Para el segundo paso del método IMPES, el cálculo explícito de las saturaciones, existe una limitación para que el sistema sea estable: el tamaño del paso de tiempo. La limitación del tamaño del paso de tiempo causada por evaluar la permeabilidad relativa en un paso de tiempo anterior, t^n, puede ser aproximada como,

$$\Delta t \leq \frac{V_{po}}{f' |uA|}, \tag{68}$$

donde uA representa la tasa de flujo volumétrico a través de un área transversal a la celda de la grilla a analizar, f' es la derivada de la curva de flujo fraccional y $V_{po} = \phi * \Delta x * \Delta y * h = \phi * \Delta x * A$ (cubo elemental). La limitación del tamaño de la grilla puede ser más severa para celdas pequeñas o cuando la tasa de flujo volumétrico transversal es grande [39].

En los sistemas con fluidos inmiscibles es suficiente realizar el análisis de la estabilidad en alguna de las fases, por ej. el agua. Asumiendo que se está trabajando con celdas equiespaciadas, la porosidad es la misma en toda la grilla y hay un solo pozo inyector de agua, podemos hacer algunas simplificaciones.

$$\frac{\sum_{i=1}^{N_X}\sum_{j=1}^{N_Y}\left[\frac{V_{b_{i,j}}\phi_{i,j}}{\alpha_c}\right]\left(S_{w_{i,j}}^{n+1} - S_{w_{i,j}}^{n}\right)}{\Delta t \sum_{i=1}^{N_X}\sum_{j=1}^{N_Y} q_{wsc_{i,j}}^{n}} = \frac{\left[\frac{V_b\phi}{\alpha_c}\right]\sum_{i=1}^{N_X}\sum_{j=1}^{N_Y} S_{w_{i,j}}^{n+1} - S_{w_{i,j}}^{n}}{\Delta t . q_{wsc_{0,0}}^{n}} \tag{69}$$

Capítulo 3

Métodos para la resolución de ecuaciones diferenciales multifásicas - El método IMPES

A diferencia de los modelos matemáticos aplicados a la resolución de simulación de reservorios de petróleo monofásicos, los sistemas multifásicos generan un sistema de múltiples ecuaciones diferenciales para cada elemento de la grilla de discretización, una ecuación para cada fase del sistema. Los métodos más usuales para la solución de este tipo de sistemas son tres: el método de soluciones simultáneas, el método IMPES y el método de soluciones secuenciales. La descripción detallada de cada método puede hallarse en [6, 14, 38]. Nuestro trabajo se concentrará en el método IMPES, dado que es el método más ampliamente utilizado en la industria.

El objetivo del método IMPES consiste en obtener una única ecuación en presión para cada punto de la grilla combinando las ecuaciones de flujo para eliminar las incógnitas de la saturación de agua. Este es el primer paso del método y se denomina implícito en presión. El sistema de ecuaciones lineales en presión resultante se puede resolver en forma directa o mediante un método iterativo para obtener el valor de la presión de petróleo en el paso de tiempo *n+1*.

El segundo paso del método consiste en resolver en forma explícita las incógnitas de la saturación de agua sustituyendo los valores de las presiones en el paso de tiempo *n+1* en una de las ecuaciones de flujo. Este paso se denomina explícito en saturación.

El método IMPES aplicado a un modelo de flujo bifásico petróleo-agua

El primer paso del método IMPES combina las ecuaciones de flujo del componente petróleo con la componente agua para eliminar la saturación de agua.

- Ecuación de flujo para el componente petróleo

$$T_{ox_{i-1/2,j}}^{n+1} p_{o_{i-1,j}}^{n+1} - \left(T_{ox_{i-1/2,j}}^{n+1} + T_{ox_{i+1/2,j}}^{n+1} + T_{oy_{i,j-1/2}}^{n+1} + T_{oy_{i,j+1/2}}^{n+1} \right) p_{o_{i,j}}^{n+1} + T_{ox_{i+1/2,j}}^{n+1} p_{o_{i+1,j}}^{n+1} + T_{oy_{i,j-1/2}}^{n+1} p_{o_{i,j-1}}^{n+1} + T_{oy_{i,j+1/2}}^{n+1} p_{o_{i,j+1}}^{n+1} = C_{op_{i,j}} \left(p_{o_{i,j}}^{n+1} - p_{o_{i,j}}^{n} \right) + C_{ow_{i,j}} \left(S_{w_{i,j}}^{n+1} - S_{w_{i,j}}^{n} \right) - q_{osc_{i,j}}^{n+1} \quad (48)$$

- Ecuación de flujo para la componente agua

$$
\begin{aligned}
&T^{n+1}_{wx_{i-1/2,j}}\, p^{n+1}_{o_{i-1,j}} - \left(T^{n+1}_{wx_{i-1/2,j}} + T^{n+1}_{wx_{i+1/2,j}} + T^{n+1}_{wy_{i,j-1/2}} + T^{n+1}_{wy_{i,j+1/2}} \right) p^{n+1}_{o_{i,j}} + \\
&T^{n+1}_{wx_{i+1/2,j}}\, p^{n+1}_{o_{i+1,j}} + T^{n+1}_{wy_{i,j-1/2}}\, p^{n+1}_{o_{i,j-1}} + T^{n+1}_{wy_{i,j+1/2}}\, p^{n+1}_{o_{i,j+1}} = \\
&C_{wp_{i,j}} \left(p^{n+1}_{o_{i,j}} - p^{n}_{o_{i,j}} \right) + C_{ww_{i,j}} \left(S^{n+1}_{w_{i,j}} - S^{n}_{w_{i,j}} \right) - q^{n+1}_{wsc_{i,j}}
\end{aligned}
\tag{49}
$$

para $i = 1, 2, ..., N_x$ y $j = 1, 2, ..., N_y$.

Como la saturación de agua, $S_{w_{i,j}}$, solo aparece en el lado derecho de las ecuaciones, multiplicando la ec. (49) por $-\dfrac{Cow_{i,j}}{Cww_{i,j}}$ y sumando el resultado a la ec. (48) se obtiene un sistema de $N_x * N_y$ ecuaciones con $N_x * N_y$ incógnitas:

$$
\begin{aligned}
&T^{n+1}_{ox_{i-1/2,j}}\, p^{n+1}_{o_{i-1,j}} - \left(T^{n+1}_{ox_{i-1/2,j}} + T^{n+1}_{ox_{i+1/2,j}} + T^{n+1}_{oy_{i,j-1/2}} + T^{n+1}_{oy_{i,j+1/2}} \right) p^{n+1}_{o_{i,j}} + \\
&T^{n+1}_{ox_{i+1/2,j}}\, p^{n+1}_{o_{i+1,j}} + T^{n+1}_{oy_{i,j-1/2}}\, p^{n+1}_{o_{i,j-1}} + T^{n+1}_{oy_{i,j+1/2}}\, p^{n+1}_{o_{i,j+1}} - \\
&\frac{C_{ow_{i,j}}}{C_{ww_{i,j}}} T^{n+1}_{wx_{i-1/2,j}}\, p^{n+1}_{o_{i-1,j}} + \frac{C_{ow_{i,j}}}{C_{ww_{i,j}}} \left(T^{n+1}_{wx_{i-1/2,j}} + T^{n+1}_{wx_{i+1/2,j}} + T^{n+1}_{wy_{i,j-1/2}} + T^{n+1}_{wy_{i,j+1/2}} \right) p^{n+1}_{o_{i,j}} - \\
&\frac{C_{ow_{i,j}}}{C_{ww_{i,j}}} T^{n+1}_{wx_{i+1/2,j}}\, p^{n+1}_{o_{i+1,j}} - \frac{C_{ow_{i,j}}}{C_{ww_{i,j}}} T^{n+1}_{wy_{i,j-1/2}}\, p^{n+1}_{o_{i,j-1}} - \frac{C_{ow_{i,j}}}{C_{ww_{i,j}}} T^{n+1}_{wy_{i,j+1/2}}\, p^{n+1}_{o_{i,j+1}} = \\
&C_{op_{i,j}} \left(p^{n+1}_{o_{i,j}} - p^{n}_{o_{i,j}} \right) + C_{ow_{i,j}} \left(S^{n+1}_{w_{i,j}} - S^{n}_{w_{i,j}} \right) - q^{n+1}_{osc_{i,j}} - \\
&\frac{C_{ow_{i,j}}}{C_{ww_{i,j}}} C_{wp_{i,j}} \left(p^{n+1}_{o_{i,j}} - p^{n}_{o_{i,j}} \right) + \frac{C_{ow_{i,j}}}{C_{ww_{i,j}}} q^{n+1}_{wsc_{i,j}}
\end{aligned}
\tag{70}
$$

que debe ser resulto en forma simultánea.

Reordenando los términos nos queda el siguiente sistema de ecuaciones:

$$
\begin{aligned}
&\left(T^{n+1}_{oy_{i,j-1/2}} - \frac{C_{ow_{i,j}}}{C_{ww_{i,j}}} T^{n+1}_{wy_{i,j-1/2}} \right) p^{n+1}_{o_{i,j-1}} + \left(T^{n+1}_{ox_{i-1/2,j}} - \frac{C_{ow_{i,j}}}{C_{ww_{i,j}}} T^{n+1}_{wx_{i-1/2,j}} \right) p^{n+1}_{o_{i-1,j}} - \\
&\left(T^{n+1}_{ox_{i-1/2,j}} + T^{n+1}_{ox_{i+1/2,j}} + T^{n+1}_{oy_{i,j-1/2}} + T^{n+1}_{oy_{i,j+1/2}} + C_{op_{i,j}} - \frac{C_{ow_{i,j}}}{C_{ww_{i,j}}} \left(T^{n+1}_{wx_{i-1/2,j}} + T^{n+1}_{wx_{i+1/2,j}} + T^{n+1}_{wy_{i,j-1/2}} + T^{n+1}_{wy_{i,j+1/2}} + C_{wp_{i,j}} \right) \right) p^{n+1}_{o_{i,j}} + \\
&\left(T^{n+1}_{ox_{i+1/2,j}} - \frac{C_{ow_{i,j}}}{C_{ww_{i,j}}} T^{n+1}_{wx_{i+1/2,j}} \right) p^{n+1}_{o_{i+1,j}} + \left(T^{n+1}_{oy_{i,j+1/2}} - \frac{C_{ow_{i,j}}}{C_{ww_{i,j}}} T^{n+1}_{wy_{i,j+1/2}} \right) p^{n+1}_{o_{i,j+1}} = \\
&-\left(C_{op_{i,j}} - \frac{C_{ow_{i,j}} C_{wp_{i,j}}}{C_{ww_{i,j}}} \right) p^{n}_{o_{i,j}} - q^{n+1}_{osc_{i,j}} + \frac{C_{ow_{i,j}}}{C_{ww_{i,j}}} q^{n+1}_{wsc_{i,j}}
\end{aligned}
\tag{71}
$$

Pero como

$$\frac{Cow}{Cww} = \frac{-\dfrac{V_b}{\alpha_c \Delta_t}\left(\dfrac{\phi}{B_o}\right)^{n+1}}{\dfrac{V_b}{\alpha_c \Delta_t}\left(\dfrac{\phi}{B_w}\right)^{n+1}} = -\left(\frac{B_w}{B_o}\right)^{n+1} \tag{72}$$

multiplicando al sistema de ecuaciones por B_o^{n+1} y reemplazando se obtiene:

$$\begin{aligned}
&\left(B^{n+1}_{o_{i,j}} T^{n+1}_{oy_{i,j-1/2}} + B^{n+1}_{w_{i,j}} T^{n+1}_{wy_{i,j-1/2}}\right) p^{n+1}_{o_{i,j-1}} + \left(B^{n+1}_{o_{i,j}} T^{n+1}_{ox_{i-1/2,j}} + B^{n+1}_{w_{i,j}} T^{n+1}_{wx_{i-1/2,j}}\right) p^{n+1}_{o_{i-1,j}} - \\
&\left(B^{n+1}_{o_{i,j}} \left(T^{n+1}_{ox_{i-1/2,j}} + T^{n+1}_{ox_{i+1/2,j}} + T^{n+1}_{oy_{i,j-1/2}} + T^{n+1}_{oy_{i,j+1/2}} + C_{op_{i,j}}\right) + B^{n+1}_{w_{i,j}} \left(T^{n+1}_{wx_{i-1/2,j}} + T^{n+1}_{wx_{i+1/2,j}} + T^{n+1}_{wy_{i,j-1/2}} + T^{n+1}_{wy_{i,j+1/2}} + C_{wp_{i,j}}\right)\right) p^{n+1}_{o_{i,j}} + \\
&\left(B^{n+1}_{o_{i,j}} T^{n+1}_{ox_{i+1/2,j}} + B^{n+1}_{w_{i,j}} T^{n+1}_{wx_{i+1/2,j}}\right) p^{n+1}_{o_{i+1,j}} + \left(B^{n+1}_{o_{i,j}} T^{n+1}_{oy_{i,j+1/2}} + B^{n+1}_{w_{i,j}} T^{n+1}_{wy_{i,j+1/2}}\right) p^{n+1}_{o_{i,j+1}} = \\
&-\left(B^{n+1}_{o_{i,j}} C_{op_{i,j}} + B^{n+1}_{w_{i,j}} C_{wp_{i,j}}\right) p^{n}_{o_{i,j}} - \left(B^{n+1}_{o_{i,j}} q^{n+1}_{osc_{i,j}} + B^{n+1}_{w_{i,j}} q^{n+1}_{wsc_{i,j}}\right)
\end{aligned} \tag{73}$$

El conjunto de ecuaciones resultante tiene la siguiente forma:

$$e_{i,j}\, p^{n+1}_{o_{i,j-1}} + a_{i,j}\, p^{n+1}_{o_{i-1,j}} + b_{i,j}\, p^{n+1}_{o_{i,j}} + c_{i,j}\, p^{n+1}_{o_{i+1,j}} + f_{i,j}\, p^{n+1}_{o_{i,j+1}} = d_{i,j} \tag{74}$$

donde

$$e_{i,j} = B^{n+1}_o T^{n+1}_{oy_{i,j-1/2}} + B^{n+1}_w T^{n+1}_{wy_{i,j-1/2}} \tag{75}$$

$$a_{i,j} = B^{n+1}_o T^{n+1}_{ox_{i-1/2,j}} + B^{n+1}_w T^{n+1}_{wx_{i-1/2,j}} \tag{76}$$

$$b_{i,j} = -\left(B^{n+1}_{o_{i,j}} \left(T^{n+1}_{ox_{i-1/2,j}} + T^{n+1}_{ox_{i+1/2,j}} + T^{n+1}_{oy_{i,j-1/2}} + T^{n+1}_{oy_{i,j+1/2}} + C_{op_{i,j}}\right) + B^{n+1}_{w_{i,j}} \left(T^{n+1}_{wx_{i-1/2,j}} + T^{n+1}_{wx_{i+1/2,j}} + T^{n+1}_{wy_{i,j-1/2}} + T^{n+1}_{wy_{i,j+1/2}} + C_{wp_{i,j}}\right)\right) \tag{77}$$

$$c_{i,j} = B^{n+1}_{o_{i,j}} T^{n+1}_{ox_{i+1/2,j}} + B^{n+1}_{w_{i,j}} T^{n+1}_{wx_{i+1/2,j}} \tag{78}$$

$$f_{i,j} = B^{n+1}_{o_{i,j}} T^{n+1}_{oy_{i,j+1/2}} + B^{n+1}_{w_{i,j}} T^{n+1}_{wy_{i,j+1/2}} \tag{79}$$

$$d_{i,j} = -\left(B^{n+1}_{o_{i,j}} C_{op_{i,j}} + B^{n+1}_{w_{i,j}} C_{wp_{i,j}}\right) p^{n}_{o_{i,j}} - \left(B^{n+1}_{o_{i,j}} q^{n+1}_{osc_{i,j}} + B^{n+1}_{w_{i,j}} q^{n+1}_{wsc_{i,j}}\right) \tag{80}$$

Este sistema de ecuaciones forma una matriz de coeficientes rala de cinco bandas no compactas.

Definición 2: La matriz de un sistema de ecuaciones lineales se denomina **rala** si solo un número pequeño de sus elementos es no nulo.

La matriz resultante no es simétrica y es estrictamente diagonal dominante. La siguiente figura muestra la estructura de la matriz obtenida.

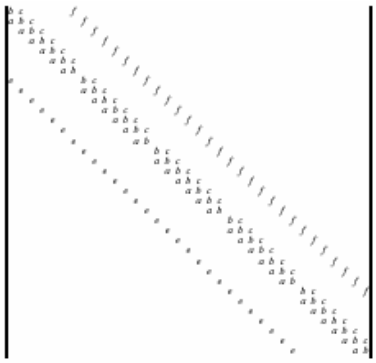

fig. 4. Matriz de 5 bandas resultante

Definición 3: La **transpuesta** de una matriz A de *m* x *n*, denotada A^t, es una matriz de *n* x *m* cuyos elementos son $\left(A^t\right)_{ij} = \left(A\right)_{ji}$. Una matriz cuya transpuesta es ella misma se llama **simétrica**.

Definición 4: Se dice que la matriz A de *n* x *n* es **estrictamente diagonal dominante** en el caso de que se satisfaga

$$\left|a_{ii}\right| > \sum_{\substack{j=1 \\ j\neq i}}^{n} \left|a_{ij}\right| \tag{81}$$

para cada i=1,2,…,*n*.

Definición 5: Se dice que una matriz A de *n* x *n* es **no-singular** si existe una matriz A^{-1} de *n* x *n* tal que $AA^{-1} = A^{-1}A = I$. La matriz A^{-1} se llama la **inversa** de A. Una matriz que no tiene inversa se llama **singular**.

Teorema 1: Para una matriz A de *n* x *n* las siguientes afirmaciones son equivalentes:

a) La ecuación $Ax = 0$ tiene una única solución ***x*** = **0**.

b) El sistema lineal $Ax = d$ tiene una única solución para cualquier vector columna ***d*** *n*-dimensional.

c) La matriz A es no-singular, es decir, A^{-1} existe.

d) $\det A \neq 0$

e) Se puede aplicar el método de eliminación de Gauss (sin intercambio de filas) al sistema lineal $Ax = d$ para cualquier vector columna ***d*** *n*-dimensional.

Teorema 2: Si A es una matriz de *n* x *n* estrictamente diagonal dominante, entonces A es **no-singular**.

Demostración: Consideremos el sistema lineal $Ax = 0$ y supongamos que existe una solución distinta de cero $x = (x_i)$ de este sistema. En este caso, para algún *k*, $0 < |x_k| = máx_{1 \le j \le n} |x_j|$. Como $\sum_{j=1}^{n} a_{ij} x_j = 0$ para cada *i* = 1,2,…, *n*, cuando *i* = *k*

$$a_{kk} x_k = -\sum_{\substack{j=1 \\ j \neq k}}^{n} a_{kj} x_j$$

Esto implica que

$$|a_{kk}||x_k| \le \sum_{\substack{j=1 \\ j \neq k}}^{n} |a_{kj}||x_j| \quad \text{o} \quad |a_{kk}| \le \sum_{\substack{j=1 \\ j \neq k}}^{n} |a_{kj}| \frac{|x_j|}{|x_k|} \le \sum_{\substack{j=1 \\ j \neq k}}^{n} |a_{kj}|$$

lo que contradice al hecho de que A sea estrictamente diagonal dominante. Por lo tanto, la única solución al sistema $Ax = 0$, es $\boldsymbol{x} = \mathbf{0}$, es decir, A es no-singular.

El Teorema 2 garantiza que el sistema de ecuaciones lineales en presión de la ec. (74) tiene solución única.

Una vez resuelto este sistema de ecuaciones, el segundo paso del método IMPES consiste en resolver las saturaciones de agua en forma explícita.

Despejando $S_{w_{i,j}}^{n+1}$ de la ec. (49) se obtiene la siguiente ecuación:

$$S_{w_{i,j}}^{n+1} = S_{w_{i,j}}^{n} + \frac{1}{C_{ww_{i,j}}} \left[\begin{array}{l} T_{wx_{i-1/2,j}}^{n+1} p_{o_{i-1,j}}^{n+1} - \left(T_{wx_{i-1/2,j}}^{n+1} + T_{wx_{i+1/2,j}}^{n+1} + T_{wy_{i,j-1/2}}^{n+1} + T_{wy_{i,j+1/2}}^{n+1} \right) p_{o_{i,j}}^{n+1} + \\ T_{wx_{i+1/2,j}}^{n+1} p_{o_{i+1,j}}^{n+1} + T_{wy_{i,j-1/2}}^{n+1} p_{o_{i,j-1}}^{n+1} + T_{wy_{i,j+1/2}}^{n+1} p_{o_{i,j+1}}^{n+1} - \\ C_{wp_{i,j}} \left(p_{o_{i,j}}^{n+1} - p_{o_{i,j}}^{n} \right) + q_{wsc_{i,j}}^{n+1} \end{array} \right] \tag{82}$$

para $i = 1, 2, ..., N_x$ y $j = 1, 2, ..., N_y$.

Resolución del sistema de ecuaciones lineales en presión

Para resolver el sistema de ecuaciones de cinco bandas no compactas que forma la ec. (74) aplicada a todos los puntos de la grilla se analizaron diferentes técnicas iterativas.

Los métodos directos para resolver sistemas de ecuaciones lineales no se adaptan muy bien cuando las matrices son ralas por las siguientes razones:

- A medida que se aplica el proceso de eliminación directa, ciertos elementos de la matriz que inicialmente eran nulos, dejan de serlo. Por lo tanto, aumenta considerablemente el

espacio de almacenamiento utilizado por la matriz. Esto se conoce con el nombre de *fill-in*.

- La mayoría de las operaciones aritméticas que realizan estos métodos se aplican a elementos nulos, desperdiciando tiempo de cómputo.

Una alternativa es utilizar métodos iterativos que realizan aproximaciones sucesivas para obtener soluciones cada vez más precisas en cada iteración. Los requerimientos de memoria para éstos son en general menores en orden de magnitud.

Métodos iterativos

En general, un método iterativo que resuelve el sistema de ecuaciones $Ax = d$ empieza con una aproximación inicial $x^{(0)}$ a la solución *x* y genera una sucesión de vectores $\left\{x^{(k)}\right\}_{k=0}^{\infty}$ que converge a la solución. La mayoría de estos métodos convierte el sistema $Ax = d$ en un sistema equivalente de la forma $x = Tx + c$. Una vez seleccionada la aproximación inicial $x^{(0)}$, la sucesión de vectores que se van aproximando a la solución se genera calculando

$$x^{(k)} = Tx^{(k-1)} + c \tag{83}$$

para cada $k = 1, 2, 3, \ldots$

La principal ventaja de los métodos iterativos para este tipo de problemas es que los requerimientos de almacenamiento son mínimos. La desventaja es que algunas técnicas iterativas, aún en las mejores circunstancias, fallan en converger [38]. La performance de los métodos iterativos se degrada a medida que aumenta el tamaño de la grilla.

Los métodos iterativos se clasifican en: estacionarios y no estacionarios. Los métodos iterativos estacionarios son los que realizan en cada iteración las mismas operaciones. Los métodos iterativos no estacionarios son aquellos que utilizan coeficientes que dependen de la iteración. En este trabajo se analizan los siguientes métodos iterativos estacionarios: Jacobi, Gauss-Seidel, Block SOR (Block Successive Overrelaxation) y SIP (Strongly Implicit Procedure).

El método iterativo de Jacobi

Dada la *i*-ésima ecuación del sistema $Ax = d$,

$$\sum_{j=1}^{n} a_{ij} x_j = d_i \tag{84}$$

Si se despeja x_i asumiendo que el resto de los coeficientes de $\boldsymbol{x}$ permanecen sin cambios, se obtiene

$$x_i = \left(d_i - \sum_{j \neq i} a_{ij} x_j \right) \Big/ a_{ii} \tag{85}$$

Esto sugiere un método iterativo definido por

$$x_i^{(k)} = \left(d_i - \sum_{j \neq i} a_{ij} x_j^{(k-1)} \right) \Big/ a_{ii} \tag{86}$$

El método requiere que $a_{ii} \neq 0$ para cada $i = 1, 2, ..., n$. Como A es estrictamente diagonal dominante, el método se puede aplicar sin inconvenientes.

El pseudo-código del método de Jacobi se describe a continuación

```
Elegir aproximación inicial x^(0)
Para k = 1, 2, ... hacer
   Para i = 1, 2, ..., n hacer
      x_i = 0
      Para j = 1, 2, ..., i-1, i+1, ..., n hacer
         x_i = x_i + a_ij x_j^(k-1)
      Fin
      x_i = (d_i - x_i) / a_ii
   Fin
   x^(k) = x
   verificar condición de corte; continuar si es necesario
Fin
```

fig. 5. El método de Jacobi

Como posibles condiciones de corte se analizaron el error absoluto

$$\left\| x^{(k)} - x^{(k-1)} \right\|_\infty < \varepsilon \tag{87}$$

y el error relativo

$$\frac{\left\| x^{(k)} - x^{(k-1)} \right\|_\infty}{\left\| x^{(k)} \right\|_\infty} < \varepsilon \tag{88}$$

con $\varepsilon > 0$.

Estas condiciones de corte se aplicaron a todos los métodos iterativos estudiados.

Definición 6: La norma l_∞ de un vector $x = \left(x_1, x_2, ..., x_n \right)^t$ se define como

$$\left\| x \right\|_\infty = \max_{1 \leq i \leq n} \left| x_i \right| \tag{89}$$

Análisis del método

Ventajas:

- Es muy fácil de implementar y paralelizar.
- Se aplica a matrices estrictamente diagonal dominante.

Desventajas:

- Converge lentamente.
- El número de iteraciones es proporcional a la cantidad de puntos en la grilla, tornándose impracticable para grillas grandes [30].

El método iterativo de Gauss-Seidel

Analizando la ec. (86) se observa que para calcular el elemento $x_i^{(k)}$ se usan los componentes de $x^{(k-1)}$. Como para $i>1$, los valores $x_1^{(k)},...,x_{i-1}^{(k)}$ ya han sido calculados y son mejores aproximaciones a la solución real $x_1,...,x_{i-1}$, si se utilizan estos valores surge el método de Gauss-Seidel,

$$x_i^{(k)} = \left(d_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k-1)} \right) \Big/ a_{ii} \tag{90}$$

Al igual que antes, el método requiere que $a_{ii} \neq 0$ para cada $i=1,2,...,n$. Como A es estrictamente diagonal dominante, el método se puede aplicar.

El pseudo-código del método de Gauss-Seidel se describe a continuación

```
Elegir aproximación inicial x^(0)
Para k = 1, 2, ... hacer
   Para i = 1, 2, ..., n hacer
      σ = 0
      Para j = 1, 2, ..., i − 1 hacer
         σ = σ + a_ij x_j^(k)
      Fin
      Para j = i + 1, ..., n hacer
         σ = σ + a_ij x_j^(k−1)
      Fin
      x_i^(k) = (d_i − σ) / a_ii
   Fin
   verificar condición de corte; continuar si es necesario
Fin
```

fig. 6. El método de Gauss-Seidel

Análisis del método

Ventajas:

- En general, converge más rápido que el método de Jacobi aunque hay excepciones.
- Se aplica a matrices estrictamente diagonal dominante.

Desventajas

- El algoritmo es de naturaleza serial aunque se puede modificar levemente coloreando los puntos de la grilla de manera tal que dos puntos adyacentes no utilicen un mismo color. Esta técnica se conoce con el nombre *red-black ordering* (ordenamiento rojo-negro), [30, 34]. Todos los puntos coloreados con un mismo color se pueden resolver en forma simultánea.

Teorema 3: Si A es estrictamente diagonal dominante, entonces para cualquier elección de $x^{(0)}$ tanto el método de Jacobi como el de Gauss-Seidel dan lugar a sucesiones $\left\{x^{(k)}\right\}_{k=0}^{\infty}$ que convergen a la solución $Ax = b$.

El método iterativo Block SOR

La técnica Block SOR (Successive Overrelaxation por Bloques) particiona el problema de resolver el sistema de ecuaciones $A * x = d$ de manera tal que los A_{ij} son submatrices; los x_i y d_i contienen subcomponentes de **x** y **d**, respectivamente.

$$\begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix} \tag{91}$$

Las submatrices de la diagonal A_{ii} son a su vez tridiagonales, para todo $i = 1, 2, ..., n$. Las submatrices $A_{i,i+1}$, para todo $i = 1, 2, ..., n-1$, y las submatrices $A_{i,i-1}$, para todo $i = 2, 3, ..., n$, son diagonales. El resto de las submatrices son nulas.

Se adaptó el siguiente algoritmo de Block SOR a este tipo de matrices para que realice sólo las operaciones en los elementos no nulos de la matriz:

$$A_{ii} y_i^{(k+1)} = -\sum_{j=1}^{i-1} A_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} A_{ij} x_j^{(k)} + d_i \,, \qquad i = 1, 2, ..., n \tag{92}$$

$$x_i^{(k+1)} = w\left\{ y_i^{(k+1)} - x_i^{(k)} \right\} + x_i^{(k)} \,, \qquad i = 1, 2, ..., n \tag{93}$$

donde $y_i^{(k+1)}$ es un vector auxiliar.

Análisis del método

Ventajas:

- El sistema de ecuaciones tridiagonal de la ec. (92) se puede resolver con el paquete LAPACK.
- Los métodos iterativos por bloques generalmente incurren en errores por redondeo menores.

Desventajas:

- El método es de naturaleza serial aunque con una leve modificación en la técnica de relajación se pueden remover las dependencias entre los datos, [16]. La tasa de convergencia puede verse afectada al modificar el orden de procesamiento.
- El factor de relajación ω es difícil de estimar.

El método iterativo SIP

El método iterativo SIP (Strongly Implicit Procedure) propuesto por [37] modifica a la matriz A, del sistema de ecuaciones (74), con una matriz N de manera tal que la factorización de la matriz $(A+N)$ tiene una mejor tasa de convergencia cuando se la utiliza en la técnica iterativa. La factorización de $(A+N)$ genera dos matrices: L y U. Ambas matrices tienen a lo sumo tres elementos no nulos en cada fila. La matriz L es una matriz triangular inferior y la matriz U es triangular superior con 1's en la diagonal principal.

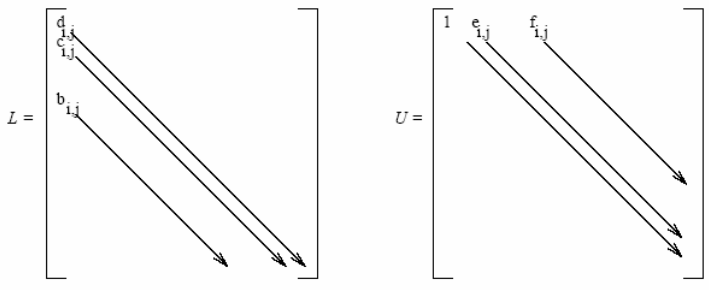

fig. 7. Formato de las matrices *L* y *U*.

La matriz $(A+N)=LU$ tiene la forma:

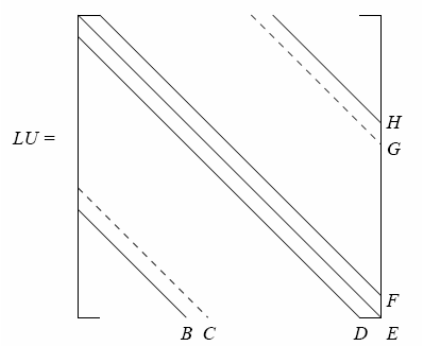

fig. 8. Formato de la matriz LU

El conjunto completo de ecuaciones que relacionan a LU con $(A+N)$ es el siguiente:

$$b_{i,j} = B_{i,j} \tag{94}$$

$$c_{i,j} = D_{i,j} \tag{95}$$

$$d_{i,j} + b_{i,j} f_{i,j-1} + c_{i,j} e_{i-1,j} = E_{i,j} \tag{96}$$

$$d_{i,j} e_{i-1,j} = F_{i,j} \tag{97}$$

$$d_{i,j} f_{i,j-1} = H_{i,j} \tag{98}$$

$$b_{i,j} e_{i,j-1} = C_{i,j} \tag{99}$$

$$c_{i,j} f_{i-1,j} = G_{i,j} \tag{100}$$

Como para cada punto (i,j) de la grilla hay cinco incógnitas, este nuevo sistema está sobredimensionado y se puede entonces ignorar las dos últimas ecuaciones.

El paso iterativo SIP para el sistema de ecuaciones $A*x=d$ se define:

$$Ax = d \tag{101}$$

$$Ax^{(k+1)} + Nx^{(k+1)} = d + Nx^{(k)}$$

$$\left(A+N\right)x^{(k+1)} = d + Nx^{(k)} + Ax^{(k)} - Ax^{(k)}$$

$$\left(A+N\right)x^{(k+1)} = d + \left(A+N\right)x^{(k)} - Ax^{(k)}; \qquad \left(A+N\right) = LU$$

$$LUx^{(k+1)} = d + LUx^{(k)} - Ax^{(k)}$$

$$LUx^{(k+1)} - LUx^{(k)} = d - Ax^{(k)}$$

$$LUx^{(k+1)} - LUx^{(k)} = d - Ax^{(k)}; \qquad \text{se define el residuo } R^{(k)} = d - Ax^{(k)}$$

$$LU\left(x^{(k+1)} - x^{(k)}\right) = R^{(k)}; \qquad \text{se define } d^{(k)} = \left(x^{(k+1)} - x^{(k)}\right)$$

$$L\left(Ud^{(k)}\right) = R^{(k)}; \qquad \text{se define } v^{(k)} = \left(Ud^{(k)}\right)$$

$Lv^{(k)} = R^{(k)} \Rightarrow v^{(k)} = L^{-1}R^{(k)}$ paso de la sustitución hacia delante

$d^{(k)} = U^{-1}v^{(k)}$ paso de la sustitución hacia atrás

se verifica la convergencia del método y se actualiza $x^{(k+1)} = d^{(k)} + x^{(k)}$.

En resumen, el algoritmo consta de cuatro pasos:

1. Se factoriza a la matriz A en una matriz L triangular inferior y una matriz U triangular superior.
2. Se sustituye hacia delante $v^{(k)} = L^{-1}R^{(k)}$
3. Se sustituye hacia atrás $d^{(k)} = U^{-1}v^{(k)}$
4. Se verifica la convergencia y se actualiza $x^{(k+1)} = d^{(k)} + x^{(k)}$.

Análisis del método

Ventajas:

- No depende de un factor de relajación.
- Converge rápidamente.

Desventajas:

- El método es de naturaleza serial aunque utilizando el ordenamiento *red-black* se puede obtener una implementación paralela eficiente [32].

LAPACK

LAPACK (Linear Algebra PACKage) es un paquete de bibliotecas escritas en Fortran 77 que se utilizan para resolver sistemas de ecuaciones lineales. Las rutinas se encuentran altamente optimizadas para lograr una alta performance en cualquier tipo de plataforma. El objetivo del proyecto LAPACK fue hacer que las bibliotecas de procesamiento numérico

(EISPACK y LINPACK) sean portables para distintas arquitecturas y estén altamente optimizadas.

Para poder invocar rutinas escritas en Fortran desde un programa escrito en C/C++ es necesario tener en cuenta lo siguiente:

- Fortran utiliza la convención *call-by-address*. C/C++ utiliza la convención *call-by-value*, lo que significa que sólo punteros deben utilizarse en la invocación a las rutinas escritas en Fortran desde un programa escrito en C/C++.
- Los arreglos en Fortran son almacenados en memoria ordenados por columnas, *column-major order*. Los arreglos en C/C++ son almacenados en memoria ordenados por filas, *row-major order*.

  Por ej., la declaración de un arreglo de enteros en C/C++ `int A[3][2]` se almacena en memoria como:

`A[0][0]`	`A[0][1]`
`A[1][0]`	`A[1][1]`
`A[2][0]`	`A[2][1]`

  En Fortran, la declaración `INTEGER A(3,2)` se almacena en memoria como:

`A(1,1)`	`A(2,1)`	`A(3,1)`
`A(1,2)`	`A(2,2)`	`A(3,2)`

La solución numérica serial

A fin de poder analizar la calidad de los resultados y la performance de la versión paralela del simulador se implementó primero una versión serial del mismo. A grandes rasgos, el algoritmo serial realiza los siguientes pasos:

1. **Lectura del archivo de configuración:** se leen los datos iniciales del problema, los parámetros PVT y las permeabilidades relativas. Dicho archivo se procesa y se inicializan las estructuras de datos correspondientes.
2. **Simulación del método IMPES:** en cada paso de tiempo se calculan los factores de volumen y las viscosidades mediante la interpolación de la presión de petróleo con los parámetros PVT; y se calculan las permeabilidades relativas mediante la interpolación de la saturación de agua. A continuación, se calculan las transmisibilidades para armar el sistema de ecuaciones lineales en presión de la ec. (74) que corresponde al primer paso del método IMPES. Para resolver este sistema se analizaron los métodos iterativos descriptos anteriormente.

Una vez resuelto el sistema, se obtienen las saturaciones en forma explícita (segundo paso del método IMPES).

3. **Evaluación de la condición de corte:** en cada paso de tiempo se calculan los caudales en el pozo extractor de petróleo:

$$Q_o = \left[\frac{k_{ro}/\mu_o}{(k_{ro}/\mu_o)+(k_{rw}/\mu_w)} \right]_{0,0} * q_{osc} \tag{102}$$

$$Q_w = \left[\frac{k_{rw}/\mu_w}{(k_{ro}/\mu_o)+(k_{rw}/\mu_w)} \right]_{0,0} * q_{osc} \tag{103}$$

Si $Q_w > 1.0$ y $Q_o = 0.0$ ó $\frac{Q_o}{Q_w} < 0.9$ se termina la simulación por que no hay más petróleo para producir.

4. **Almacenamiento de los resultados:** los datos calculados se almacenan en un archivo con el formato NetCDF. NetCDF (Network Common Data Form) es una interfaz para el acceso y almacenamiento de datos científicos en formato matricial. Los resultados se analizan luego con el programa MatLab. Para esto fue necesario instalar MexCDF, que es una interfaz entre MatLab y NetCDF para manipular los archivos en dicho formato.

Capítulo 4

Paralelización del problema

Por qué paralelizar?

Una computadora paralela se define como un conjunto de elementos de procesamiento que cooperan entre sí de tal manera que resuelven problemas de mayor envergadura respecto de sus pares secuenciales [15]. La necesidad de tener mayor poder de cómputo ha sido la principal causa para el desarrollo de las computadoras. La computación paralela le ha permitido a los científicos obtener un poder de cómputo virtualmente ilimitado.

En la vida real, cuando necesitamos que una tarea se realice más rápido las opciones son:

- trabajar más duro
- trabajar de manera inteligente
- pedir ayuda

Desde el punto de vista computacional, su análogo sería:

- utilizar un procesador más veloz
- desarrollar un algoritmo más eficaz
- utilizar más procesadores

Al aumentar el número de procesadores, se espera poder realizar más trabajo en menos tiempo. Para que estos procesadores cooperen entre sí es necesario:

1. decidir e implementar una red de intercomunicación para los procesadores y los módulos de memoria,
2. diseñar e implementar el software necesario para este hardware,
3. idear algoritmos y estructuras de datos para resolver nuestro problema,
4. dividir los algoritmos y estructuras de datos en subproblemas,
5. identificar las comunicaciones que serán necesarias entre los distintos subproblemas y
6. asignar los subproblemas a los distintos procesadores y módulos de memoria.

Estos son los problemas críticos que se presentan en el desarrollo de la computación paralela como alternativa efectiva a la computación serial [29]. En este trabajo nos concentramos en los ítems 3-6.

Una de las clasificaciones de los sistemas paralelos es la llamada SIMD y MIMD. SIMD se refiere a las siglas en inglés de *Single Instruction Multiple Data*. SIMD es una técnica que se utiliza para aumentar el poder de procesamiento de las computadoras. Un sistema SIMD puro tiene una CPU (unidad central de procesamiento) y una gran colección de ALUs (Arithmetic

Logical Unit) subordinadas. La idea en estos sistemas es la de aplicar la misma instrucción a distintos datos, o por extensión, un mismo programa aplicado a distintos datos. MIMD se refiere a las siglas en inglés *Multiple Instruction Multiple Data*. MIMD contrasta con los sistemas SIMD en el sentido de que hay múltiples procesadores autónomos donde cada uno maneja su propio conjunto de datos. A su vez, los sistemas paralelos MIMD (Multiple Instruction Multiple Data) pueden dividirse en sistemas con memoria compartida (multiprocesadores) y en sistemas con arquitectura de memoria distribuida (multicomputadores). En estos sistemas, los procesadores son autónomos, trabajan de manera asincrónica y necesitan comunicarse entre sí para intercambiar datos. Ambas topologías se encuentran disponibles en el Departamento de Computación de la Facultad de Ciencias Exactas y Naturales, UBA. El equipo multiprocesador, llamado *Sócrates*, es un Sun UltraSparc. El equipo de multicomputadores es un cluster Beowulf, llamado *Speedy Gonzalez*, y está compuesto por 16 nodos heterogéneos. Un cluster Beowulf es una computadora paralela de alta performance que está compuesta por un conjunto de PCs interconectadas por una red privada de alta velocidad. En un cluster Beowulf, cada computadora o procesador se denomina "nodo". Los nodos del *cluster* están dedicados a correr procesos paralelos y no necesariamente son máquinas homogéneas. Los *clusters* son una alternativa de bajo costo a las supercomputadoras tradicionales.

Dos grupos se han encargado de desarrollar estándares para programar en sistemas paralelos. El primer grupo, llamado High Performance Fortran Group, ha desarrollado un conjunto de extensiones para Fortran 90 que permite a los programadores escribir programas donde se paralelizan los datos (data-parallel program). La idea es la siguiente: los datos se distribuyen entre los procesadores y cada uno de éstos le aplica las mismas operaciones a su porción de datos. El segundo grupo, llamado Message-Passing Interface (MPI) Forum, en lugar de desarrollar un nuevo lenguaje de programación, ha especificado una biblioteca de funciones que pueden ser llamadas desde un programa hecho en C o Fortran. Esta biblioteca contiene un pequeño grupo de funciones que pueden ser utilizadas para lograr paralelismo mediante el intercambio de mensajes (message passing). Las funciones permiten transmitir datos de un proceso a otro [29]. Un *proceso* es la unidad conceptual fundamental de un software paralelo. Algunas de las ventajas de MPI que vale la pena resaltar son las siguientes:

- existen varias implementaciones portables y gratuitas de MPI como por ejemplo MPICH [27],
- es totalmente portable,
- está formalmente especificado y

- es un estándar.

Cuando múltiples procesadores cooperan para realizar una tarea, se genera un cierto overhead o sobrecarga que surge de tratar que todas las partes trabajen correctamente. Similarmente, cuando un grupo de *n* programadores trabajan juntos, no se obtiene como resultado un trabajo realizado *n* veces más rápido.

Medición de la performance

Al realizar un trabajo de paralelización hay muchos factores que influyen negativamente en los tiempos de ejecución de un algoritmo paralelo. Algunos de estos factores son:

- Sincronización e intercambio de mensajes entre los distintos procesos.
- Partición del problema en subdominios.
- Tiempos de espera de un proceso (cuando necesita datos de un proceso vecino para poder continuar).
- Hay partes que necesariamente deben ser ejecutadas en forma serial.

Ley de Amdahl

En el año 1967, Amdahl [2] señaló que en todo proceso de paralelización de un algoritmo serial existe una fracción de ese algoritmo que necesariamente deberá seguir ejecutándose en forma serial. Amdahl terminó con la creencia de que al agregar más procesadores para resolver un problema, su ejecución sería más rápida.

Sin embargo, en la práctica, a medida que el tamaño del problema aumenta, la fracción que es inherentemente serial disminuye.

La principal razón que motiva a la programación paralela es obtener una mejor performance, ya sea para resolver problemas más rápido que en una computadora serial o para resolver problemas más grandes que antes no podían ser resueltos.

Poder medir la performance de un programa en paralelo es una manera de estimar cuán eficientes han sido nuestros esfuerzos para poder dividir una aplicación en módulos que cooperen entre sí en paralelo.

Speedup

La medida de performance más visible es el *tiempo de ejecución*. Al medir cuánto tiempo necesita un programa en paralelo para resolver nuestro problema, estamos midiendo directamente su efectividad. Para poder determinar cuán mejor es un programa paralelo con su par serial se toma la relación entre el tiempo necesario para resolver el problema en forma

serial y el tiempo necesario para resolverlo en forma paralela. Esta relación se conoce con el nombre de *speedup* (aceleración).

Se define entonces *speedup* de un programa paralelo corriendo en *p* procesadores a:

$$S(n, p) = \frac{T1(n)}{Tp(n, p)} \tag{104}$$

donde,

$T1(n)$: tiempo necesario para la ejecución de un programa serial

$Tp(n, p)$: tiempo necesario para la ejecución de la versión paralela del mismo programa con *p* procesadores.

Cuando $S(n, p) = p$ se denomina *speedup lineal*. El speedup lineal es el óptimo desde el punto de vista de la paralelización, puesto que indica una proporción directa entre el tiempo de ejecución y la cantidad de procesadores que intervienen. Cuanto mayor sea el speedup, mejor será la performance de un algoritmo paralelo.

Eficiencia y overhead

El segmento de programa que es inherentemente serial puede ser paralelizado de dos maneras: un proceso puede ejecutar las sentencias de ese segmento mientras que el resto permanece *idle* (en espera) o todos los procesos pueden ejecutar las sentencias, es decir, las sentencias del segmento son replicadas por todos los procesos. Ambas soluciones producen *overhead* paralelo: aumentan el total de trabajo que realiza el programa paralelo sobre el total de trabajo que realiza el programa serial [29].

La eficiencia de un algoritmo paralelo mide el aprovechamiento de recursos y se define como:

$$E(n, p) = \frac{S(n, p)}{p} = \frac{T1(n)}{Tp(n, p).p} \tag{105}$$

Para el caso óptico de *speedup* lineal, $E(n, p) = 1$, es equivalente a decir que “la cantidad de trabajo realizado por el programa paralelo es el mismo que la cantidad de trabajo realizado por el programa serial”.

El *overhead* (o sobrecarga) se define entonces como la cantidad de trabajo realizado por el programa paralelo que no es realizado por el programa serial.

$$Overhead = Tp(n, p).p - T1(n) \tag{106}$$

Hay tres principales causas de *overhead*:

1. la comunicación entre procesos,
2. el tiempo que pasa un proceso esperando información de otro proceso y

3. la realización de cálculos extra que no estaban presentes en el programa serial.

Intercambio de mensajes

Un programa paralelo puede ser diseñado de dos maneras:

1. **paralelizando los datos (data-parallel approach):** en este diseño se particionan los datos entre los distintos procesos. Cada proceso ejecuta más o menos el mismo conjunto de instrucciones sobre sus datos.
2. **paralelizando el control (control-parallel approach):** en este diseño se particionan las tareas a realizar entre los procesos. Cada proceso ejecuta instrucciones que son diferentes a las instrucciones de algunos o de todos los demás procesos.

La opción de paralelizar mediante la partición del dominio de los datos es la comúnmente más utilizada. Los programas suelen ser más escalables, es decir, pueden ser utilizados para resolver problemas más grandes con más y más procesos.

En este trabajo se utilizó MPI para la parelización del simulador. MPI utiliza una mezcla de ambas técnicas, pero la paralelización del dominio de los datos suele predominar [29]. Nos hemos focalizado en implementar una solución utilizando MPI, lo que lleva a adoptar necesariamente el modelo MIMD [29].

En el modelo de intercambio de mensajes, *message-passing*, los procesos coordinan sus actividades enviando y recibiendo explícitamente mensajes. Las primitivas básicas de comunicación son *send* (enviar) y *receive* (recibir). A cada proceso se le asigna un número de rango entre $0,1,...p-1$, donde p es el número de procesos. El programador debe tener en mente *dónde* están los datos, decidir *cuándo* comunicarse con otros procesos, con *quién* se debe comunicar y *qué* se debe comunicar. Todo esto hace que la programación con el paradigma de intercambio de mensajes sea difícil y propensa a errores, pero no imposible.

La primitiva *send* se utiliza para enviar un mensaje, que contiene datos, desde un proceso a otro. El envío puede ser bloqueante o no bloqueante. Cuando se utiliza una primitiva bloqueante, por ej. `MPI_Send`, el proceso que envía información a un proceso vecino no puede continuar corriendo hasta que el receptor le confirme que ha recibido la información. El proceso emisor debe permanecer ocioso hasta que el proceso receptor esté listo para recibir la información. Cuando un proceso utiliza una primitiva no bloqueante, por ej. `MPI_Isend`, continúa con el procesamiento luego de despachar el mensaje.

La primitiva *receive* se utiliza para recibir un mensaje de datos que fue enviado por otro proceso. La recepción de datos también puede ser bloqueante o no bloqueante. Cuando se

utiliza una primitiva bloqueante, por ej. `MPI_Recv`, el proceso debe esperar a que se reciba el mensaje para poder continuar con la ejecución. Cuando se utiliza una primitiva no bloqueante, por ej. `MPI_Irecv`, el proceso le indica al sistema que está dispuesto a recibir información sin tener que bloquearse a la espera de la misma.

Las primitivas no bloqueantes le permiten a los procesos seguir realizando otras tareas solapando la comunicación de los datos con cómputos locales. Estas primitivas constan de dos partes: las funciones que le indican al sistema que un proceso desea intercambiar información con otro proceso y las funciones que verifican que el intercambio haya sido realizado. Las funciones para verificar que un pedido ha finalizado son `MPI_Test` y `MPI_Wait`. La función `MPI_Test` se utiliza para verificar que se ha completado una operación no bloqueante. La operación `MPI_Wait` no le devuelve el control al proceso hasta que finalice la operación de intercambio de información.

Cabe señalar que, por más que la comunicación no sea bloqueante, no necesariamente es asincrónica. La utilización de primitivas no bloqueantes requiere del chequeo de la disponibilidad de los datos cuando éstos requieren ser utilizados, agregando una complejidad extra al momento de implementar.

La estrategia de paralelización

Para crear un programa paralelo es necesario realizar los siguientes pasos [5]:

- identificar el trabajo que puede ser realizado en paralelo
- particionar el trabajo y los datos entre los diferentes procesos
- manejar el acceso a los datos, la comunicación y la sincronización

La comunicación y la sincronización entre los procesos es muy costosa. Para lograr una buena performance en un algoritmo paralelo hay que reducir ambas al máximo y esto se logró mediante un refinamiento sucesivo.

A modo de ejemplo, supongamos que trabajamos con comunicación bloqueante y que el proceso *P1* desea enviarle ciertos datos al proceso *P2*. Cuando el *P1* le indica al sistema que va a enviarle datos a *P2*, mediante un llamado a la función `MPI_Send`, *P1* debe esperar a que *P2* esté listo para recibirlos. Recién cuando *P2* le indica que está listo, mediante un llamado a la función `MPI_Recv`, se produce el envío de la información. Si la velocidad de los procesadores es muy diferente y un procesador va más adelantado que el otro, los tiempos de sincronización son aún mayores.

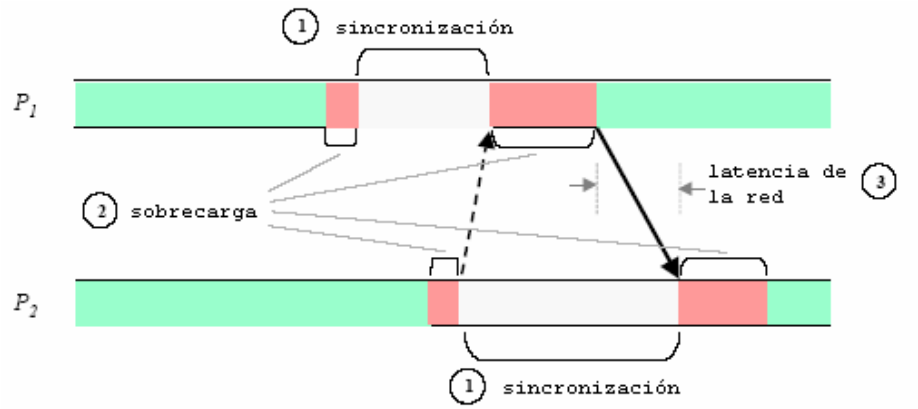

fig. 9. Costos de la comunicación entre procesos

Balance de carga

Cuando se trabaja con procesos heterogéneos es necesario implementar alguna técnica de balance de carga para evitar que los procesos más lentos se conviertan en el cuello de botella.

Al realizar las pruebas de performance, antes de introducir el balance de carga, se notó que los tiempos de comunicación y sincronización entre los procesos eran muy elevados. En este trabajo se decidió implementar el balance semi-dinámico propuesto en [26] para aprovechar mejor los recursos del cluster. Al comienzo del algoritmo se analiza el poder de cómputo de cada proceso y en base a esto se le asigna una determinada cantidad de columnas de la grilla a cada uno. A los procesos más veloces se le asignan más columnas ya que pueden realizar más trabajo en menos tiempo.

Los resultados que se obtuvieron luego de balancear la carga fueron muy alentadores. Los tiempos de comunicación y sincronización se redujeron considerablemente. Cabe señalar que se agregó una complejidad extra en las soluciones paralelas al realizar un balance de carga al principio de la ejecución que no estaba presente en la solución serial.

Paralelización del método IMPES

El objetivo de desarrollar un simulador de reservorio de petróleo que corra en máquinas paralelas es para poder realizar simulaciones en modelos de reservorios más grandes. Estas simulaciones requieren mucho tiempo de cómputo por la complejidad de sus cálculos. Una opción para acelerar las simulaciones consiste en dividir el problema en segmentos más pequeños y realizar las simulaciones en procesadores paralelos.

En esta sección se detalla el método utilizado en la paralelización del método IMPES y el diseño del programa paralelo. Inicialmente se diseñó e implementó una solución que utiliza sólo primitivas bloqueantes, para luego diseñar e implementar una solución que utiliza primitivas no bloqueantes.

La estrategia de paralelización del método IMPES

Los objetivos principales de la paralelización son: reducir el tiempo total de procesamiento y trabajar sobre problemas más grandes que antes no podían ser resueltos. La regla general para cumplir con estos objetivos es que la cantidad de cómputos por tarea paralela debe ser grande en relación a la cantidad de datos que deben ser transportados para realizar esa tarea [13].

Se analizaron varias formas de particionar los datos entre los distintos procesos. Como el reservorio es de dos dimensiones, existen varias técnicas para particionar la grilla del dominio. Se puede particionar el conjunto de datos:

- en una dimensión: por filas o por columnas
- en ambas dimensiones: por filas y por columnas
- particionando el dominio con el ordenamiento por coloreo *red-black*.

Una forma sencilla de distribuir una grilla entre un conjunto de procesos es mediante una distribución por bloques de filas o columnas. A cada proceso se le asigna un bloque consecutivo de celdas. De esta manera, los procesos sólo deben intercambiar datos con su vecino izquierdo y derecho, salvo de los casos de borde que se intercambia información con un único vecino. De esta manera se reduce notoriamente la comunicación entre procesos. En este trabajo se decidió partir y distribuir la grilla en rebanadas de bloques de columnas. Cada rebanada se ejecuta en un proceso diferente.

A modo de ejemplo, supongamos que tenemos una grilla de $7x9$ elementos y contamos con tres procesos para realizar los cálculos. La forma de partir el dominio de datos, en forma equitativa, por columnas resulta en asignarle al proceso 0 las tres primeras columnas, al proceso 1 las tres siguientes y al proceso 2 las tres columnas restantes.

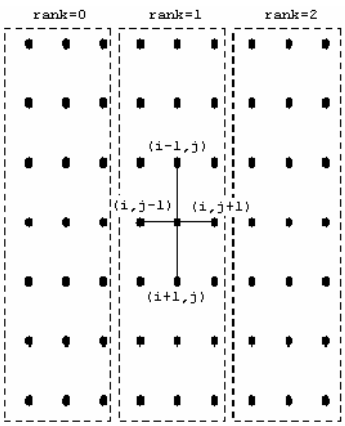

fig. 10. Distribución del dominio en rebanadas

Como los procesos deben intercambiar información para poder realizar sus cálculos locales, es necesario que cada uno de ellos pueda realizar la mayor cantidad de procesamiento en forma local. Para el cálculo de las transmisibilidades, cada celda de la grilla necesita los datos de sus vecinos. Por ej. la celda (i, j) necesita los datos de las celdas vecinas: $(i, j-1)$, $(i, j+1)$, $(i-1, j)$ y $(i+1, j)$; ver fig. 3. A cada proceso se le asignó una columna más por cada proceso vecino, estas columnas reciben el nombre de columnas fantasma [5]. Así el proceso 0 tiene los datos de la primer columna del subdominio del proceso 1. El proceso 1 tiene la última columna del subdominio del proceso 0 y la primer columna del subdominio del proceso 2. Por último, el proceso 2 tiene la última columna del subdominio del proceso 1. El subdominio sobre el cual trabajará cada proceso es el que se muestra en la fig. 11.

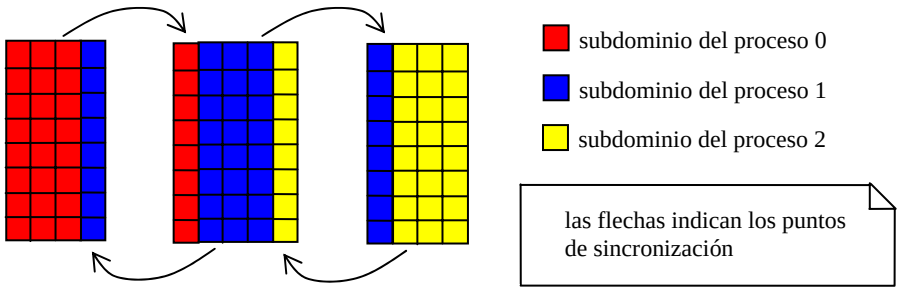

fig. 11. Distribución de la grilla entre los procesos

En cada paso de tiempo del método IMPES, los procesos deben sincronizar las columnas de los bordes para actualizar los valores de la presión de petróleo y la saturación de agua a los procesos vecinos. Para este intercambio, resultó útil agrupar y enviar la presión de petróleo y la saturación de agua en un mismo mensaje que realizar dos intercambios: primero la presión de petróleo y luego la saturación de agua. El *overhead* (sobrecarga) de la comunicación se amortiza con las grandes transferencias de datos [5].

En la partición del dominio con el ordenamiento por coloreo *red-black*, se consideran parte del mismo dominio todas las celdas de un mismo color. Dos celdas adyacentes no se pueden colorear con un mismo color. En la figura siguiente se muestra un esquema de una partición *red-black* con las columnas fantasma para aligerar los tiempos de comunicación entre los bordes de los dominios.

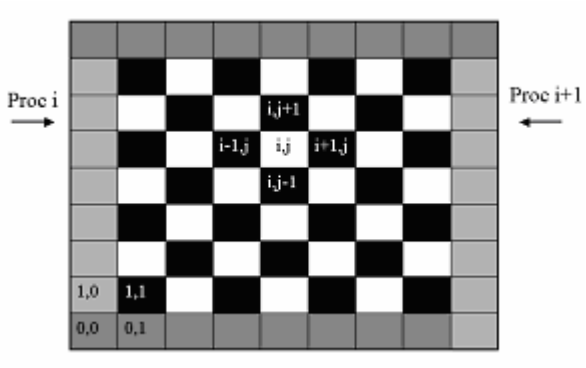

fig. 12. Partición del dominio con ordenamiento *red-black*

En este trabajo, dado que el método iterativo seleccionado (SIP) requiere una partición de tipo *red-black* se implementó una combinación de la partición por bloques de columnas y a su vez, subparticionando los bloques con el ordenamiento por coloreo *red-black.*

Una vez resuelto el problema de dividir el dominio de datos, cada proceso es responsable de realizar los cálculos en su respectivo dominio. Cada proceso sabe qué identificador de proceso le fue asignado y la cantidad total de procesos que hay corriendo en el sistema. De esta manera cada uno sabe a quién le tiene que enviar información y de quién espera recibirla.

Se eligió al proceso 0 como *coordinador*. Este proceso lee el archivo de configuración donde se encuentran los datos iniciales del problema, los parámetros PVT y las permeabilidades relativas; y le envía estos datos al resto de los procesos.

Cada proceso se encarga de almacenar los datos calculados en su dominio en el formato NetCDF. Esta información se utiliza al final de los cálculos para visualizar los resultados.

Cada proceso calcula la presión inicial del petróleo y la saturación de agua para cada celda de su subdominio (incluyendo las celdas que se comparten entre los vecinos).

Mientras haya petróleo para producir, cada proceso realiza los siguientes cálculos en cada una de las celdas de su subdominio:

- Obtiene la saturación de petróleo (en base a la saturación de agua).
- Obtiene la presión de agua en base a la presión de petróleo y la presión capilar (en nuestro caso, la presión capilar se consideró nula).
- Interpola los parámetros PVT (el factor de volumen y la viscosidad de los fluidos) con la presión de petróleo.
- Interpola las permeabilidades relativas con la saturación de agua.
- Calcula los coeficientes de las ec. 44, 45, 46 y 47.
- Calcula las transmisibilidades del petróleo (ec. 48) y del agua (ec. 49) en ambas coordenadas.

- Con los datos calculados, cada proceso arma su parte del sistema de ecuaciones del primer paso del método IMPES y resuelve el sistema de ecuaciones con un método iterativo, por ej. SIP.
- Obtiene la saturación de agua, segundo paso del método IMPES.
- Los procesos sincronizan la saturación de agua y la presión de petróleo de las celdas compartidas.
- El proceso *coordinador* analiza la condición de corte verificando el caudal de extracción de petróleo de la celda donde está ubicado el pozo productor y le comunica al resto de los procesos la decisión de continuar o terminar con la explotación del reservorio.

Paralelización de los métodos iterativos

Jacobi y Gauss Seidel

Tanto el método iterativo de Jacobi como el método de Gauss-Seidel no se implementaron en su versión paralela. Al analizar los resultados de la versión serial, no demostraron ser de gran utilidad ya que la tasa de convergencia depende, en última medida, de la dimensión de la grilla [30].

Block SOR

Para la paralelización del método Block SOR se utilizó la idea descripta en [25]. Un método de relajación se define determinando una forma de actualizar los valores en un subdominio utilizando los valores recientemente calculados de los subdominios adyacentes y un orden en el cual los subdominios van a ser procesados. Aunque la tasa de convergencia puede verse afectada por el orden de procesamiento, no es obligatorio utilizar un orden específico. Por lo tanto, como las dependencias de los datos dependen del orden de procesamiento, muchas de estas dependencias pueden ser removidas con un reordenamiento apropiado. El costo que se paga es una posible reducción en la tasa de convergencia.

Para poder paralelizar el algoritmo, se subdividió el dominio de cada proceso en filas de bloques. Cada *fila-bloque* corresponde a N_X filas consecutivas de la matriz A. Simultáneamente, cada proceso realiza los cómputos por cada *fila-bloque* de su subdominio. Mientras el proceso 0 resuelve el sistema de la *fila-bloque* 0, el proceso 1 hace lo mismo con el sistema de la *fila-bloque* 0 de su subdominio. Cuando un proceso calcula todos los valores de la iteración $k+1$, sincroniza con sus vecinos los valores de borde. De esta manera, se remueven las dependencias entre los bloques contiguos y pueden ser procesados en paralelo.

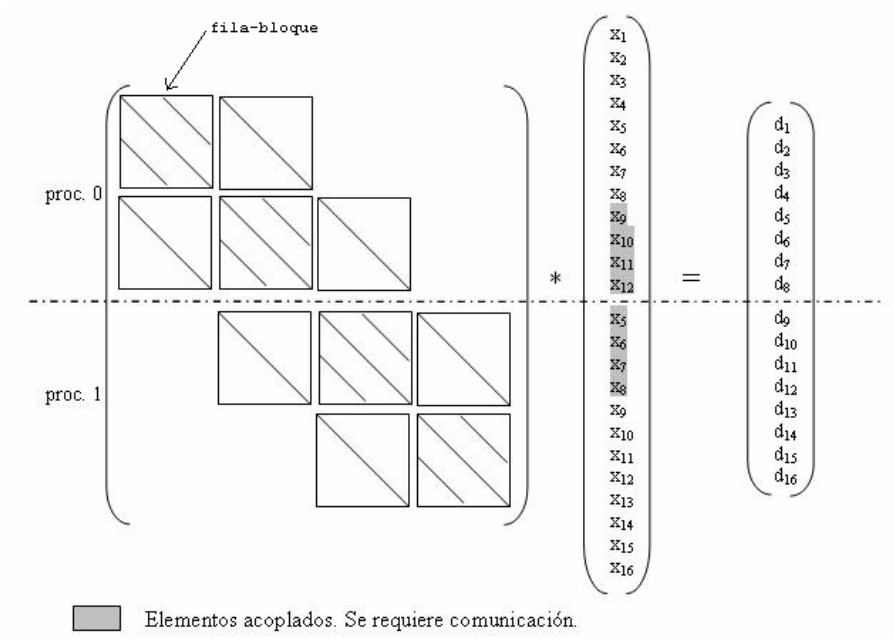

fig. 13. Paralelización del método BSOR

En la iteración $k+1$ del método BSOR, se arma con cada *fila-bloque* el sistema de la ec. 92 y se resuelve utilizando funciones del paquete LAPACK. Luego, se resuelve la ec. 93 y con estos valores se arma el vector $x^{(k+1)}$.

Para verificar la convergencia del método, cada proceso computa el error relativo *local*. Todos estos errores locales deben ser combinados en un error *global*, es decir, el error relativo entre $x^{(k)}$ y $x^{(k+1)}$, usando una función de reducción que provee MPI.

Se puede lograr una mejora de la performance si no se verifica la convergencia del método en cada iteración, dado que el costo de la función de reducción es $O(\log(p))$, donde p es el número de procesos. Al no invocar a la función de reducción en cada iteración:

1. El método puede converger entre dos llamadas a dicha función por lo que se estarían realizando cálculos innecesarios.
2. En la práctica, se notó que el llamado a la función de reducción servía además como función de sincronización entre los procesos. Al estar sincronizados los procesos, se pudo observar que disminuían notablemente los tiempos de envío y recepción de mensajes.

Verificar convergencia cada x iteraciones	Tiempo de ejecución (segs)
2	3836
3	3459

fig. 14. Grilla 21x21 sobre sobre procesadores con comunicación bloqueante.

Al principio el método iterativo Block SOR resultó muy atractivo por su velocidad de convergencia y porque permitía utilizar el paquete LAPACK cuyas funciones se encuentran altamente optimizadas. Pero al realizar algunas mediciones de performance se notó que los tiempos de comunicación y sincronización del método iterativo eran muy elevados. Este problema nos obligó a introducir el balance de carga semi-dinámico descripto anteriormente. El parámetro de sobrerrelajación w pasó a depender de la matriz A y de cómo estaban distribuidas las columnas de A entre los procesos. El balance de la carga generó otro problema: la determinación de w no se podía hacer *a priori*. Se necesitaba un método que no dependiera de un factor de ajuste. El método iterativo SIP fue el elegido.

SIP

Para la paralelización del método SIP se utilizó la idea descripta en [32]. El análisis de la versión serial de SIP muestra los siguientes grafos de dependencias para los pasos 1 a 3:

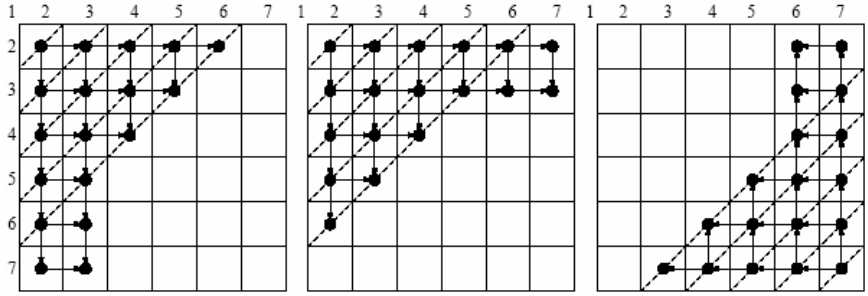

fig. 15 Paso 1: factorización LU. Paso 2: sustitución hacia delante. Paso 3: sustitución hacia atrás.

El ordenamiento *red-black* no se puede aplicar al paso 1 pero sí se puede aplicar a los pasos 2 y 3. Estos dos últimos pasos se pueden implementar realizando dos pasadas. En la primer pasada, todos los puntos de color rojo se pueden actualizar en forma simultánea. En la segunda pasada, todos los puntos de color negro se actualizan en forma simultánea. Al final de cada pasada, los procesos intercambian sus datos.

El paso 4 se implementó igual que en el método BSOR.

La paralelización del método iterativo SIP se realizó sin complicaciones. El método ha demostrado ser útil y eficiente para resolver problemas de gran tamaño. Se pudo lograr un buen paralelismo al utilizar el método de ordenamiento *red-black*.

Primitivas de comunicación

Comunicación bloqueante

Inicialmente se utilizaron las primitivas básicas de comunicación: `MPI_Send` para enviar y `MPI_Recv` para recibir. Sin embargo, la función `MPI_Send` se considera *unsafe* (insegura) ya que depende de que el sistema utilice *buffers* para almacenar los datos que se deben enviar. En el modo de comunicación *buffered*, es el proceso y no el sistema el que se debe encargar de proveer los buffers necesarios para el envío de datos. Cuando se inicia un envío y el receptor aún no ha indicado que está listo para recibir los datos, el proceso que envía *debe* almacenar los datos en forma local.

Las funciones de comunicación que se utilizaron en la versión bloqueante para el primer paso de IMPES fueron:

- `MPI_Buffer_attach`: crea el buffer de almacenamiento de datos. Sólo puede haber un único buffer.
- `MPI_Buffer_detach`: libera el buffer de almacenamiento.
- `MPI_Bsend`: envía los datos.
- `MPI_Recv`: recibe los datos.

Para el segundo paso del método IMPES se utilizaron las primitivas de comunicación bloqueantes más simples: `MPI_Send` y `MPI_Recv`. No hubo necesidad de utilizar primitivas más complejas en este paso.

Comunicación no bloqueante

Los procesos deben sincronizarse al final de cada iteración del método iterativo. Esto impide que se pueda solapar la comunicación con el procesamiento como se pensó inicialmente que iba a suceder. Sin embargo, como los procesos se comunican siempre con los mismos vecinos para intercambiar los valores de borde, se analizó la utilización de primitivas de comunicación persistente con el objetivo de reducir la sobrecarga, *overhead*, de la comunicación.

Las primitivas de comunicación persistente se pueden utilizar tanto para enviar como para recibir. En la comunicación convencional con primitivas no bloqueantes para enviar un dato

hay que llamar a la función `MPI_Isend` y luego `MPI_Wait` para esperar que termine de enviarse el mensaje. Con la comunicación persistente hay que llamar a las funciones:

- `MPI_Send_init`: para crear un objeto del tipo *request*. Aquí se indica a qué proceso se le desea enviar información.
- `MPI_Start`: tiene el mismo efecto que llamar a `MPI_Isend` pero sin el overhead de la comunicación ya que se reduce el encabezado del mensaje.
- `MPI_Wait`: espera que termine de enviarse el mensaje.
- `MPI_Request_free`: libera el objeto *request*.

La función `MPI_Send_init` se llama antes de que empiece el método iterativo y la función `MPI_Request_free` se llama cuando termina el método iterativo. Es decir, se llaman sólo una vez.

Al reducir la sobrecarga de la comunicación entre procesos, se pudo observar una pequeña disminución en los tiempos de procesamiento.

Para el segundo paso del método IMPES se utilizaron las primitivas de comunicación no bloqueantes más simples: `MPI_Isend` y `MPI_Irecv`. Aquí tampoco hubo necesidad de utilizar primitivas más complejas.

Capítulo 5

Resultados

En esta sección se muestran los resultados obtenidos para distintas simulaciones. Se compara la versión serial con las versiones paralelas (bloqueante y no bloqueante). Se incluyen algunas imágenes de los resultados obtenidos con las simulaciones y se analizan los tiempos de ejecución para obtener conclusiones sobre la performance de las implementaciones paralelas.

El simulador desarrollado se aplica para analizar el flujo bifásico petróleo-agua hacia un pozo productor para un conjunto de datos típicos tomados de [14]. El reservorio es bidimensional. Está discretizado en una grilla de $4x4$ celdas, consta de un pozo productor ubicado en la celda $(1,1)$ y un pozo inyector ubicado en la celda $(4,4)$. Se inyecta agua en forma constante a 90 barriles/día y se extrae petróleo en forma constante a 75.96 barriles/día.

Estos valores se utilizaron para verificar el correcto funcionamiento del simulador.

Constante	**Valor**	**Descripción**
Δt	10	Paso de tiempo, se mide en días.
N_X	4	Cantidad de puntos de la grilla en la dirección *x*.
N_Y	4	Cantidad de puntos de la grilla en la dirección *y*.
Q_o	-75.96	Tasa de producción de petróleo, se mide en barriles por día.
Q_w	90.0	Tasa de inyección de agua, se mide en barriles por día.
$S_{w_{INI}}$	0.12	Saturación inicial de agua.
$p_{o_{INI}}$	1000.0	Presión inicial de petróleo, se mide en psi.
r_w	0.328	Radio del pozo, se mide en ft.
r_e	1000.0	Radio externo de drenaje, ft.
h	30	Espesor de la capa productiva, ft.
k_x	0.3	Permeabilidad en la dirección *x*, se mide en Darcy.
k_y	0.3	Permeabilidad en la dirección *y*, se mide en Darcy.
ϕ	0.2	Porosidad de la roca.
TOL	$1x10^{-12}$	Tolerancia.
Δ_x	250.0	Tamaño de una celda de la grilla en la dirección *x*, ft.
Δ_y	250.0	Tamaño de una celda de la grilla en la dirección *y*, ft.

Parámetros PVT

p_o (psi)	B_o	B_w	μ_o (cp)	μ_w (cp)
14.7	1.062	1.0	1.04	0.5
264.7	1.15	1.0	0.975	0.5
514.7	1.207	1.0	0.91	0.5
1014.7	1.295	1.0	0.83	0.5
2014.7	1.435	1.0	0.695	0.5
2514.7	1.5	1.0	0.641	0.5
3014.7	1.565	1.0	0.594	0.5
4014.7	1.695	1.0	0.51	0.5
5014.7	1.827	1.0	0.556	0.5
9014.7	1.579	1.0	0.74	0.5

Permeabilidades relativas

S_w	k_{rw}	k_{ro}
0.12	0.0	1.0
0.191	0.0051	0.999
0.25	0.0102	0.800
0.294	0.0168	0.7241
0.357	0.0275	0.6206
0.414	0.0424	0.504
0.490	0.0665	0.317
0.557	0.097	0.3020
0.630	0.1148	0.1555
0.673	0.1259	0.0956
0.719	0.1381	0.0576
0.789	0.1636	0.0

La fig. 16 muestra los resultados obtenidos con la simulación realizada con el programa serial utilizando los datos del reservorio arriba enumerados. Los gráficos representan la variación en la presión de petróleo en función del tiempo. En la fig. 16.a se muestra el reservorio en condiciones iniciales ($p_{o_{INI}} = 1000\ psi$). A medida que se extrae petróleo en el punto $(1,1)$ de la grilla, su presión comienza a disminuir, tal como se ve en la fig. 16.b después de 100 días de extracción a caudal constante. En el otro extremo de la grilla, en el punto $(4,4)$, se inyecta agua para aumentar la presión de petróleo y poder así extraer más. Como se puede apreciar, la presión de petróleo va disminuyendo a medida que pasan los días. La disminución de la presión de petróleo es mayor cerca del pozo productor. Luego de 7220 días se termina la explotación del reservorio porque no hay más petróleo que se pueda extraer del pozo.

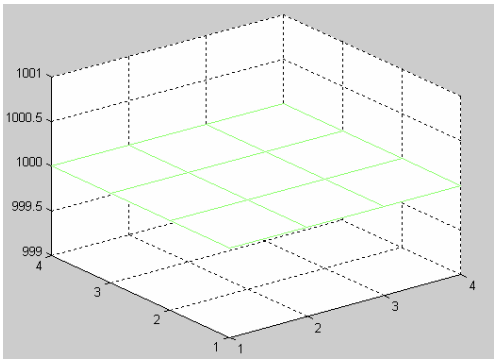

fig. 16.a $t = 0$

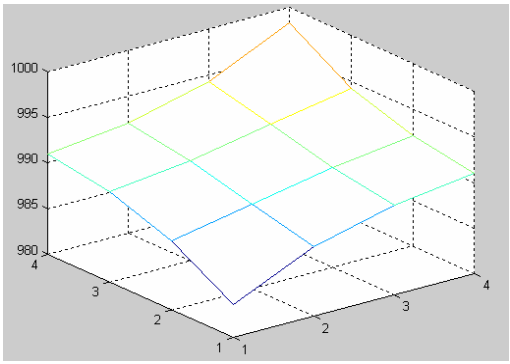

fig. 16.b $t = 100$ días

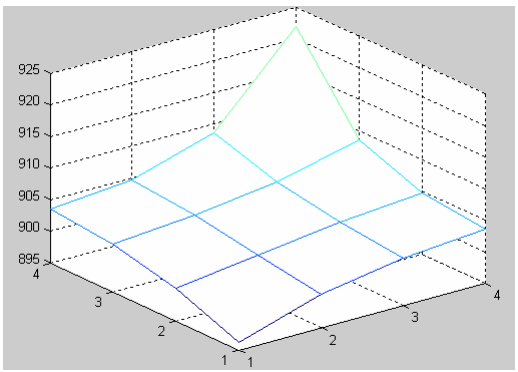

fig. 16.c $t = 1000$ días

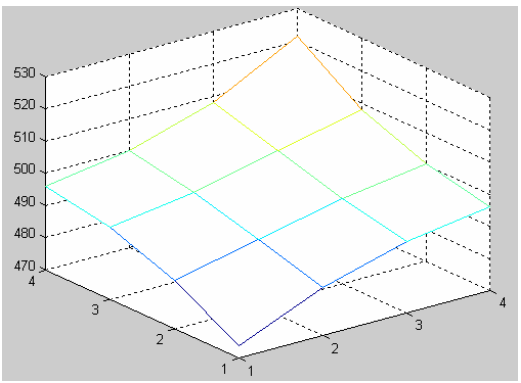

fig. 16.d

$t = 7220$ días (fin de la explotación)

En la siguiente figura se muestra la variación de la presión de petróleo en función del tiempo en la celda que contiene al pozo productor cuando se extraen 75.96 barriles/día a caudal constante. Como el caudal de extracción es constante, la presión de petróleo declina lentamente.

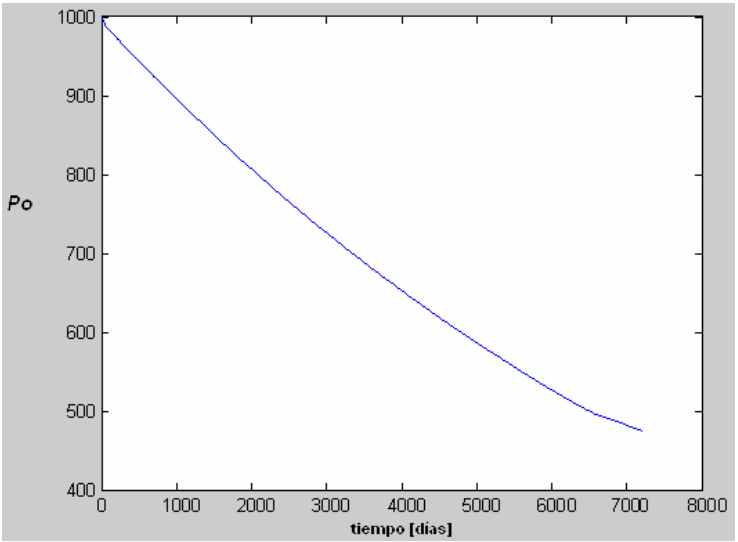

fig. 17 Presión de petróleo en la celda que contiene al pozo productor

La fig. 18 muestra como se va saturando de agua el reservorio a medida que van pasando los días. El agua se inyecta agua a caudal constante (90 barriles/días) en la celda $(4,4)$. El agua se escurre al resto de las celdas del reservorio.

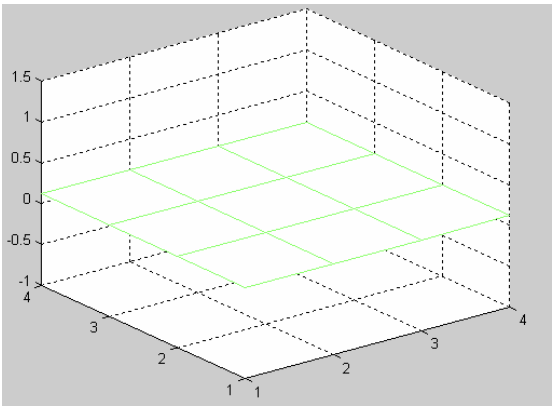

fig. 18.a $t = 0 \;\; S_{w_{INI}} = 0.12$

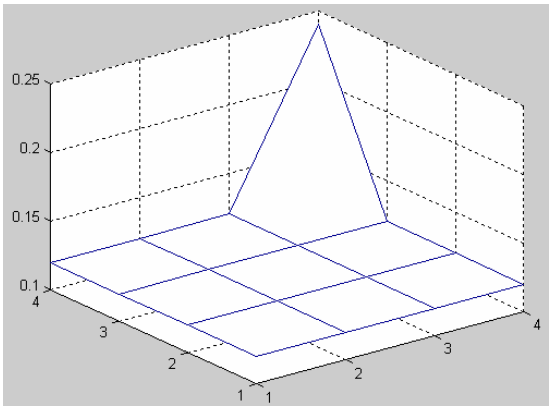

fig. 18.b $t = 100$ días

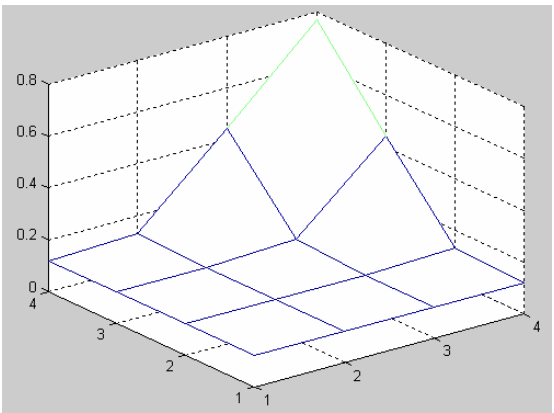

fig. 18.c $t = 1000$ días

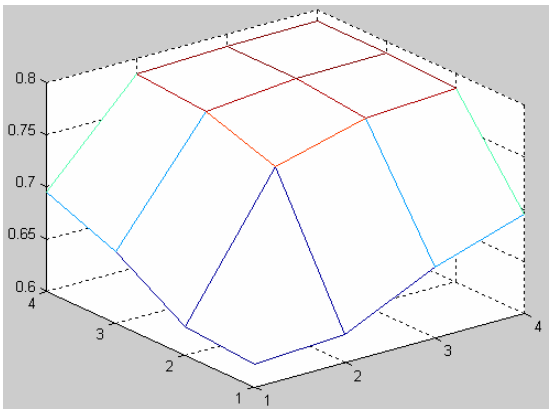

fig. 18.d

$t = 7220$ días (fin de la explotación)

En la fig. 19 se grafica la saturación de petróleo en la celda $(0,0)$ que contiene al pozo productor en función del tiempo. Cuando el agua que se inyecta en la celda $(4,4)$ empieza a escurrirse hacia la celda $(0,0)$, dicha celda comienza a saturarse de agua.

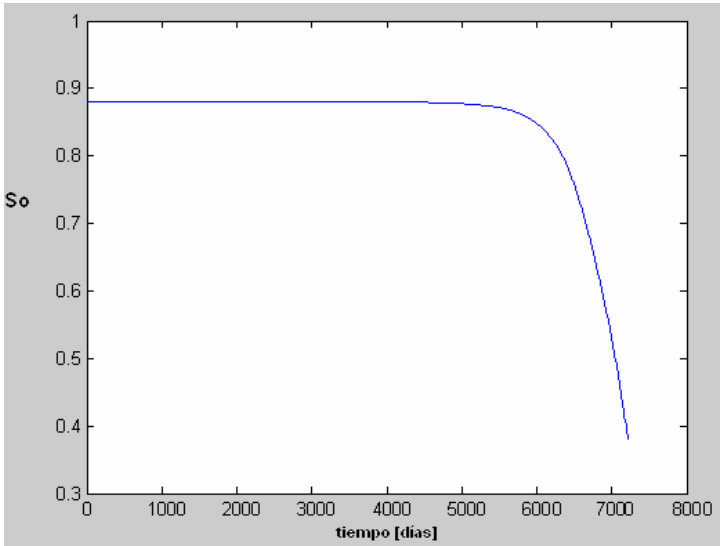

fig. 19 Saturación de petróleo en la celda que contiene al pozo productor

Análisis de tiempos y performance

En esta sección se muestran los tiempos obtenidos en las distintas corridas seriales y paralelas. Con estos resultados se analiza el speedup, la eficiencia y la sobrecarga en todas las simulaciones paralelas.

Las versiones paralelas se ejecutaron en el cluster Beowulf *Speedy Gonzalez* de 16 máquinas heterogéneas y en un Sun UltraSparc de 64 bits, llamado *Sócrates*, con dos procesadores (cada uno de 1 Ghz y con 2 GB de memoria compartida), para comparar la performance en dos tipos de arquitecturas paralelas: una con memoria distribuida conectada por una red dedicada y la otra con memoria compartida.

Para verificar la validez de los simuladores paralelos se utilizó el programa *ncdiff*. El programa *ncdiff* permite comparar dos archivos NetCDF restando las variables que poseen el mismo nombre en ambos archivos y almacena los resultados en un tercer archivo. Con los resultados del nuevo archivo generado se pudo corroborar el correcto funcionamiento de los simuladores paralelos.

Como no disponemos de datos de reservorios reales más grandes, los datos que se presentan a continuación se obtuvieron de [18] y fueron adaptados para nuestro tipo de problema.

Los parámetros de entrada que se utilizaron para analizar la performance son los siguientes:

- grilla 50x50: se obtuvo una producción de 492 días.

Constante	Valor	Descripción
Δt	1	Paso de tiempo, se mide en días.
N_X	50	Cantidad de puntos de la grilla en la dirección *x*.
N_Y	50	Cantidad de puntos de la grilla en la dirección *y*.

Q_o	-20.0	Tasa de producción de petróleo, se mide en barriles por día.
Q_w	25.0	Tasa de inyección de agua, se mide en barriles por día.
$S_{w_{INI}}$	0.12	Saturación inicial de agua.
$p_{o_{INI}}$	1500.0	Presión inicial de petróleo, se mide en psi.
r_w	0.1	Radio del pozo, ft.
r_e	820.0	Radio externo de drenaje, ft.
h	30	Espesor de la capa productiva, ft.
k_x	0.2	Permeabilidad en la dirección *x*, se mide en Darcy.
k_y	0.2	Permeabilidad en la dirección *y*, se mide en Darcy.
ϕ	0.2	Porosidad de la roca.
TOL	$1x10^{-11}$	Tolerancia.
cge	5	Verificar convergencia del método iterativo cada *x* iteraciones
Δ_x	16.4	Tamaño de una celda de la grilla en la dirección *x*, ft.
Δ_y	16.4	Tamaño de una celda de la grilla en la dirección *y*, ft.

La fig. 20 muestra los resultados obtenidos con la simulación realizada con el programa serial utilizando los datos del reservorio arriba enumerados. Los gráficos representan la variación en la presión de petróleo en función del tiempo. En la fig.20.a se muestra el reservorio en condiciones iniciales ($p_{o_{INI}} = 1498.992\ psi$). A medida que se extrae petróleo en el punto $(1,1)$ de la grilla, su presión comienza a disminuir, tal como se ve en la fig.20.b después de 100 días de extracción a caudal constante (20 barriles/día). En el otro extremo de la grilla, en el punto $(50,50)$, se inyecta agua (25 barriles/día) para aumentar la presión de petróleo. El petróleo va disminuyendo a medida que pasan los días. Luego de 492 días se termina la explotación del reservorio porque no hay más petróleo que se pueda extraer del pozo.

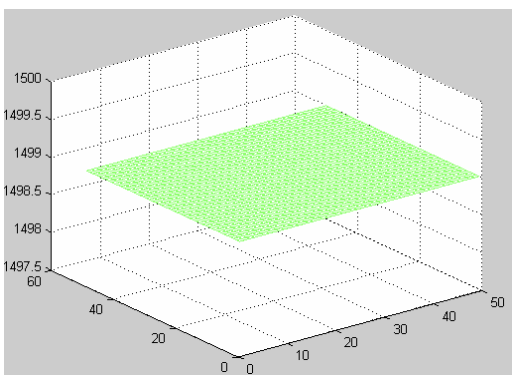

fig. 20.a $t = 0$

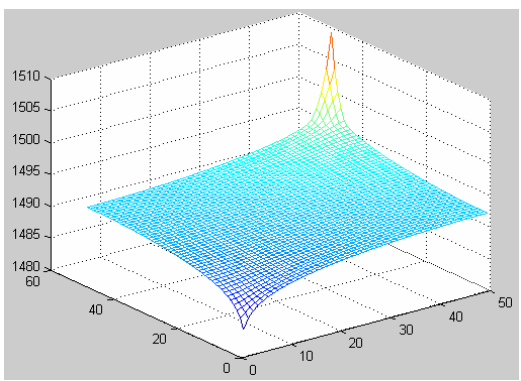

fig. 20.b $t = 100$ **días**

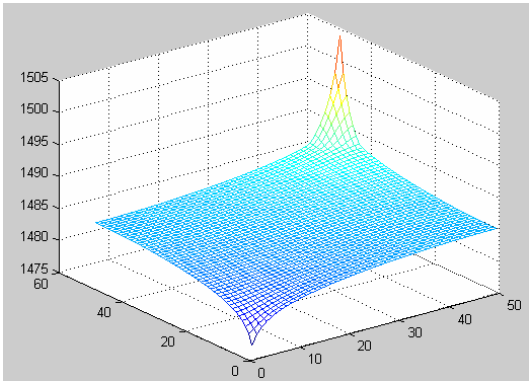

fig. 20.c $t = 200$ **días**

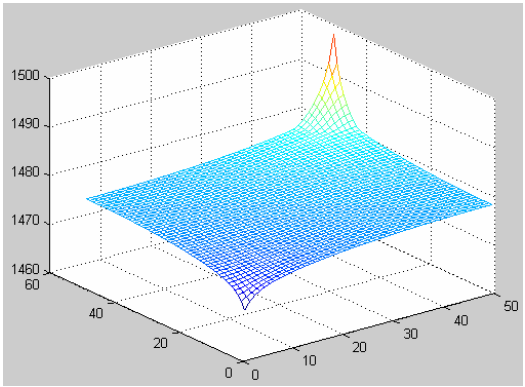

fig. 20.d $t = 300$ **días**

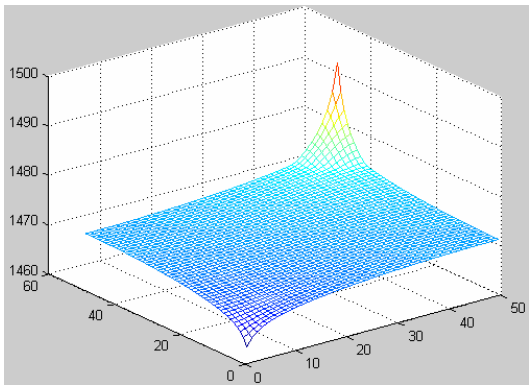

fig. 20.e $t = 400$ **días**

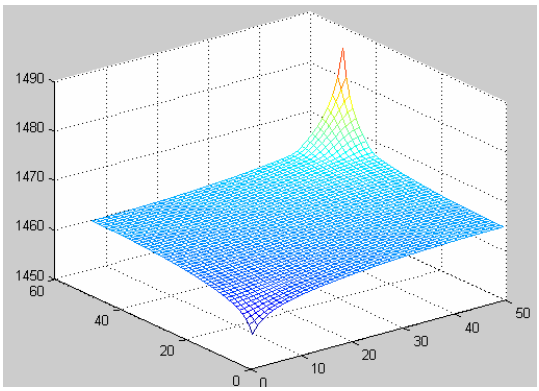

fig. 20.f $t = 492$ **días**

fin de la explotación

- grilla 100x100: se obtuvo una producción de 301 días.

Constante	**Valor**	**Descripción**
Δt	1	Paso de tiempo, se mide en días.
N_X	100	Cantidad de puntos de la grilla en la dirección *x*.
N_Y	100	Cantidad de puntos de la grilla en la dirección *y*.
Q_o	-20.0	Tasa de producción de petróleo, se mide en barriles por día.
Q_w	25.0	Tasa de inyección de agua, se mide en barriles por día.
$S_{w_{INI}}$	0.12	Saturación inicial de agua.
$p_{o_{INI}}$	1500.0	Presión inicial de petróleo, se mide en psi.
r_w	0.1	Radio del pozo, ft.
r_e	1280.0	Radio externo de drenaje, ft.
h	30	Espesor de la capa productiva, ft.
k_x	0.2	Permeabilidad en la dirección *x*, se mide en Darcy.
k_y	0.2	Permeabilidad en la dirección *y*, se mide en Darcy.
ϕ	0.2	Porosidad de la roca.
TOL	$1x10^{-11}$	Tolerancia.

cge	3	Verificar convergencia del método iterativo cada *x* iteraciones
Δ_x	12.8	Tamaño de una celda de la grilla en la dirección *x*, ft.
Δ_y	12.8	Tamaño de una celda de la grilla en la dirección *y*, ft.

Comparación de los tiempos de ejecución

No todos los nodos del cluster Speedy Gonzalez estuvieron conectados cuando se realizaron las pruebas. Como máximo hubo hasta 6 nodos habilitados en un mismo momento: 3 nodos Pentium III de 733 MHz, 1 nodo AMD Athlon de 946 Mhz y 2 nodos AMD Athlon de 900 Mhz. Como se puede observar, los nodos del cluster son bien heterogéneos.

Las primeras mediciones de performance realizadas en el cluster fueron muy desalentadoras. Los tiempos de comunicación y sincronización eran altísimos. A modo de ejemplo, se muestran a continuación los tiempos de corrida de la versión paralela no bloqueante para la grilla de $50x50$ en 4 procesadores:

	Tiempo de ejecución (en segs)
Sin balance de carga	5402
Con balance de carga	4815

fig. 21. Ejemplo de la mejora obtenida al realizar un balanceo de carga en el cluster de PC.

Los factores que afectaron la performance fueron varios:

- Verificación de la convergencia del método iterativo: la frecuencia con la que se verifica la convergencia del método iterativo y la forma en que se la verifica afectaron la performance de las soluciones con primitivas paralelas. Verificar la performance en todas las iteraciones resultó muy costoso. Por otro lado, la función `MPI_AllReduce` resultó más costosa que llamar a la función `MPI_Reduce`, concentrar los resultados en el proceso *coordinador* y que éste le comunique al resto de los procesos si el algoritmo ha convergido a la solución. Cuando se utilizaban más de 2 procesos, los tiempos de sincronización de la función `MPI_AllReduce` eran muy costosos.

  A continuación se presentan las dos opciones. En la primer opción: si hay que verificar la convergencia en esa iteración (`iter%cge==0`), entonces todos los procesos calculan las normas infinito y le envían los resultados al coordinador. Si el error relativo es menor que cierta tolerancia, el coordinador le avisa al resto de los procesos que deben terminar de iterar.

En la segunda opción: si hay que verificar la convergencia en esa iteración, entonces todos los procesos calculan las normas infinito y se ponen de acuerdo entre ellos para ver si el método ha convergido. Todos toman la decisión de seguir o no iterando.

```
. . .
double maxs[2];
if (iter % cge==0) {
   maxs[0] = fabs(x[offset]-x0[offset]);
   maxs[1] = fabs(x[offset]);
   for (int i=1; i<rows; i++) {
       maxs[0] = MAX(maxs[0], fabs(x[i+offset]-x0[i+offset]));
       maxs[1] = MAX(maxs[1], fabs(x[i+offset]));
   }
}
. . .
if (iter % cge==0) {
   double infinityNorms[2];
   MPI_Reduce(maxs,    infinityNorms,    2,    MPI_DOUBLE,    MPI_MAX,    0,
   MPI_COMM_WORLD);
   short int resp = 0;
   if (rank == 0) {
       double relativeError = infinityNorms[0] / infinityNorms[1];
       if (relativeError < TOL) {
           resp = 1;
       }
   }

   MPI_Bcast(&resp, 1, MPI_SHORT, 0, MPI_COMM_WORLD);
   if (resp) {
       foundSolution = true;
       break; //Solution found!
   }
}
. . .
```

fig. 22.a Utilizando `MPI_Reduce` y `MPI_Bcast`.

```
. . .
double maxs[2];
if (iter % cge==0) {
   maxs[0] = fabs(x[offset]-x0[offset]);
   maxs[1] = fabs(x[offset]);
   for (int i=1; i<rows; i++) {
       maxs[0] = MAX(maxs[0], fabs(x[i+offset]-x0[i+offset]));
       maxs[1] = MAX(maxs[1], fabs(x[i+offset]));
   }
}
. . .
if (iter % cge == 0) {
   double error;
   double infinityNorms[2];
   MPI_Allreduce(&maxs,&infinityNorms,2,MPI_DOUBLE,MPI_MAX,MPI_COMM_WORLD);

   error = infinityNorms[0] / infinityNorms[1];
   if (error < TOL) {
       foundSolution = true;
       break; //Solution found!
   }
}
. . .
```

fig. 22.b. Utilizando `MPI_AllReduce`.

- Balance de carga: al principio no se estaba realizando ningún balance de carga sobre los nodos del cluster. Los procesadores del mismo son muy heterogéneos y los nodos más lentos retrasaban al resto. Luego de introducir el balance de carga

semi-dinámico se notó una mejora considerable en la performance pudiendo lograr un speedup muy cercano al óptimo.

- Primitivas de comunicación: el uso de primitivas de comunicación más complejas ayudó a mejorar la performance. Cuando se analizaron las primitivas de comunicación bloqueante se optó por el modo de comunicación *buffered* y cuando se analizaron las primitivas no bloqueantes se optó por la comunicación persistente.

Los tiempos de corrida, medidos en segundos, se resumen a continuación. En la fig. 23.a. se presentan los tiempos de ejecución de las corridas realizadas sobre el cluster Speedy Gonzalez. En la fig. 23.b. se presentan los tiempos de ejecución de las corridas realizadas sobre Sócrates.

Grilla ***n x n***	**Speedy Gonzalez**						
	Número de procesadores						
		1	**2**	**3**	**4**	**5**	**6**
50x50	**bloq**	18564	10890	7536	5187	---	---
	no-bloq		10680	7198	4815	---	---
100x100	**bloq**	70313	43250	31266	24422	18463	15111
	no-bloq		43121	30526	23669	17976	14702

fig 23.a. Tiempos de corrida en el cluster

Grilla ***n x n***	**Sócrates**		
	Número de procesadores		
		1	**2**
50x50	**bloq**	18587	10216
	no-bloq		10216
100x100	**bloq**	84534	46299
	no-bloq		46274

fig 23.b. Tiempos de corrida en Sócrates

De la tabla de la fig. 23.a se puede observar:

- La grilla de $50x50$ es muy pequeña para utilizar más de 4 procesadores. Los nodos del cluster reciben pocas columnas de la grilla cuando se supera ese número. La comunicación se hace más importante que los cómputos locales.
- A medida que se aumenta el número de procesos del cluster, la reducción en los tiempos de ejecución es menos notoria. Hay ciertas partes del programa que deben ejecutarse secuencialmente y no pueden paralelizarse.

Dado que en la versión paralela no bloqueante no se pudieron solapar en gran medida los cálculos locales con la comunicación, no se obtuvo una ventaja con el uso de estas primitivas.

El uso de memoria compartida en Sócrates hace que la velocidad de transferencia entre los dos procesos sea despreciable. En todas las pruebas realizadas en Sócrates, el balance de carga repartió el trabajo en forma equitativa (50-50).

Los tiempos de ejecución obtenidos en el cluster de PC nos muestran que este tipo de arquitectura es una alternativa de bajo costo frente a los supercomputadores tradicionales.

Speedup

Como primer medida de la performance, se analiza el speedup obtenido en las distintas corridas. Se toma el cociente entre el tiempo de ejecución de la versión serial y el tiempo de ejecución de la versión paralela. La fig. 24.a. muestra la aceleración obtenida en el cluster Speedy Gonzalez. La fig. 24.b. muestra la aceleración obtenida en Sócrates.

Grilla ***n x n***	**Speedy Gonzalez**					
	Número de procesadores					
		2	**3**	**4**	**5**	**6**
50x50	**bloq**	1.70	2.46	3.58	---	---
	no-bloq	1.74	2.58	3.86	---	---
100x100	**bloq**	1.63	2.25	2.88	3.81	4.65
	no-bloq	1.63	2.30	2.97	3.91	4.78

fig 24.a. Speedup obtenido en el cluster

Grilla ***n x n***	**Sócrates**	
	Número de procesadores	
		2
50x50	**bloq**	1.82
	no-bloq	1.82
100x100	**bloq**	1.83
	no-bloq	1.83

fig 24.b. Speedup obtenido en Sócrates

Con estos resultados, se presentan a continuación los gráficos con la comparación entre la aceleración ideal (speedup lineal) y las aceleraciones obtenidas con las diferentes configuraciones en el cluster Speedy Gonzalez.

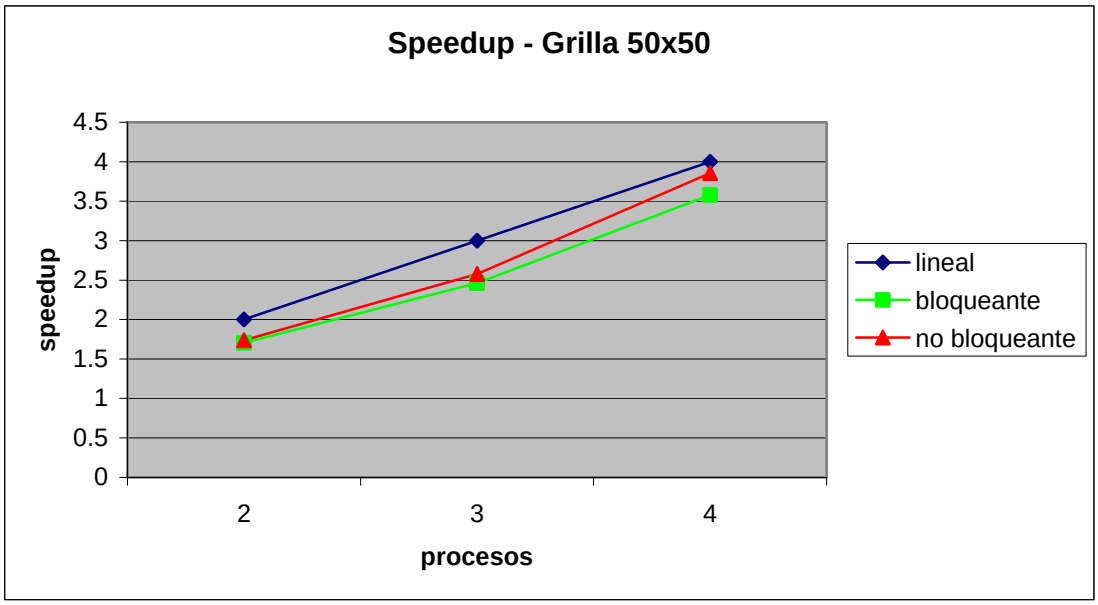

fig. 25. Speedup de la Grilla 50x50 en el cluster Speedy Gonzalez

En la fig. 25 se puede apreciar una mejora al utilizar primitivas de comunicación no bloqueante.

Al aumentar el tamaño de la grilla también aumenta la cantidad de iteraciones del método iterativo y por consiguiente aumenta la comunicación y la cantidad de datos que se transmiten entre los procesos, tal como se observa en la fig. 26. Por lo tanto, el speedup para esta configuración se alejó del óptimo.

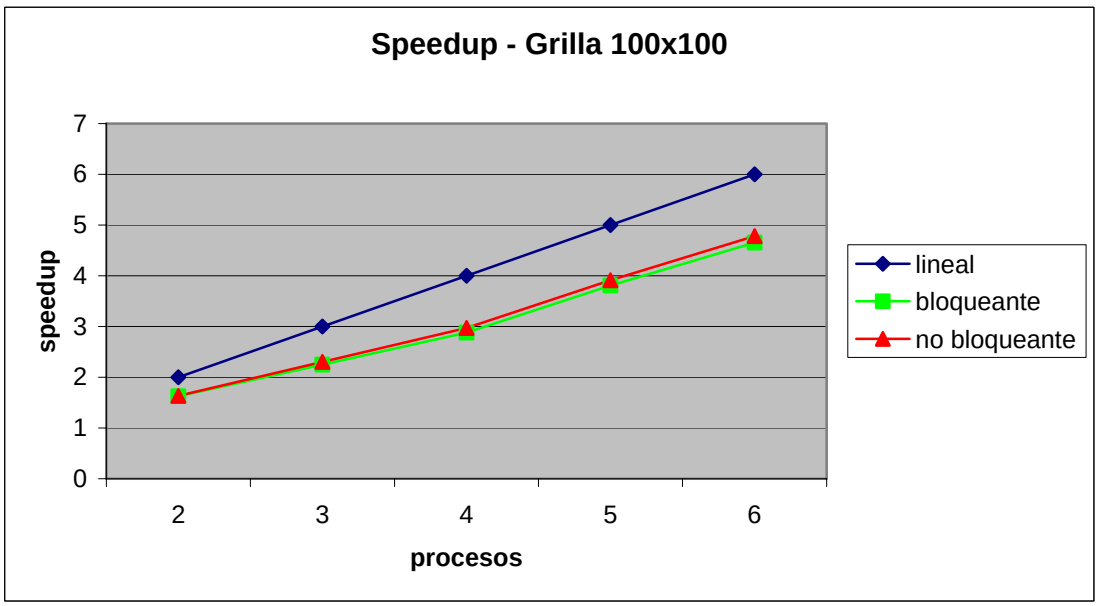

fig. 26. Speedup de la Grilla 100x100 en el cluster Speedy Gonzalez

En las pruebas realizadas en el cluster se puede observar una mejora en la performance cuando se utilizan primitivas de comunicación no bloqueante. Los recursos se aprovechan mejor al solapar comunicación con procesamiento local lográndose una reducción en el tiempo total de ejecución.

En las pruebas realizadas en Sócrates, el speedup está muy cercano al lineal tal como se observa en la fig. 27. No hubo diferencias entre la performance del algoritmo bloqueante con respecto al algoritmo no bloqueante.

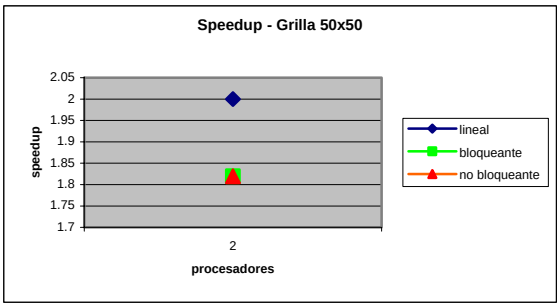

fig. 27.a. Speedup de la grilla de 50x50 en Sócrates

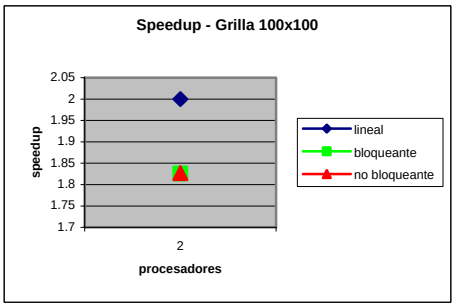

fig. 27.b. Speedup de la grilla de 100x100 en Sócrates

Eficiencia

La segunda medición de relevancia es la eficiencia del algoritmo paralelo para las mismas corridas del caso anterior. Se toma el cociente entre el speedup y la cantidad de procesos. Los resultados obtenidos se presentan a continuación. La fig. 28.a. presenta la eficiencia obtenida en el cluster de PC. La fig. 28.b. muestra la eficiencia obtenida en la máquina con dos procesadores con memoria compartida. Como se puede apreciar de los resultados, las mediciones realizadas en Sócrates muestran una eficiencia muy cercana a la óptima.

Grilla ***n x n***	**Speedy Gonzalez**					
	Número de procesadores					
		2	**3**	**4**	**5**	**6**
50x50	**bloq**	0.85	0.82	0.89	---	---
	no-bloq	0.87	0.86	0.96	---	---
100x100	**bloq**	0.81	0.75	0.72	0.76	0.78
	no-bloq	0.82	0.77	0.74	0.78	0.80

fig 28.a. Eficiencia obtenida en el cluster

Grilla ***n x n***	**Sócrates**	
	Número de procesadores	
		2
50x50	**bloq**	0.91

	no-bloq	0.91
100x100	**bloq**	0.91
	no-bloq	0.91

fig 28.b. Eficiencia obtenida en Sócrates

Con estos resultados, se presentan a continuación los gráficos con la comparación entre la eficiencia lineal y las eficiencias obtenidas con las diferentes configuraciones en el cluster de PC.

Las implementaciones no bloqueantes resultaron más eficientes que las versiones bloqueantes.

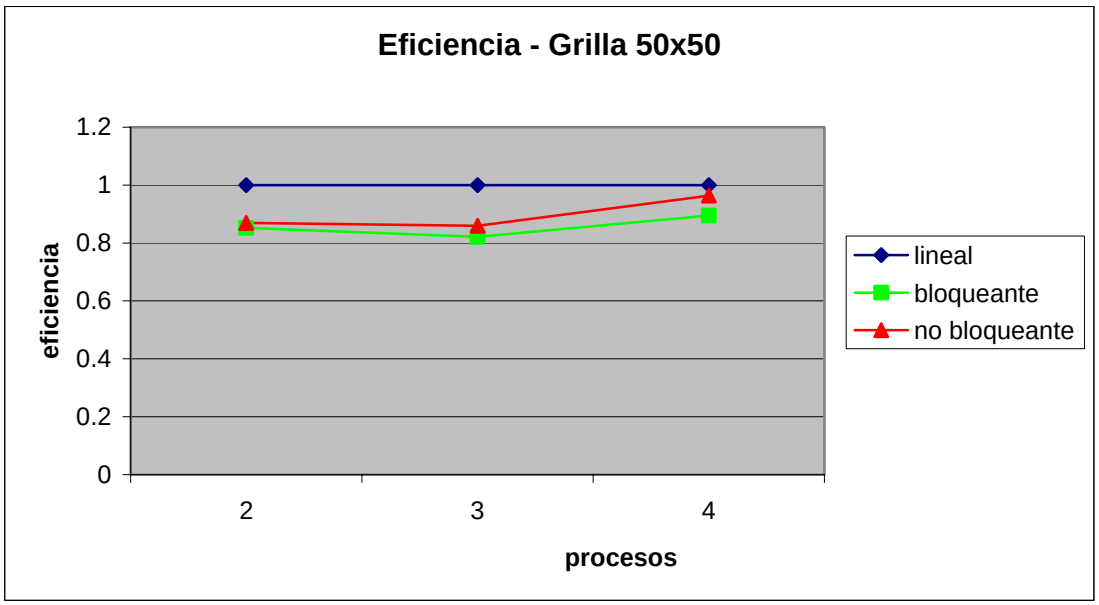

fig. 29. Eficiencia de la Grilla 50x50 en el cluster Speedy Gonzalez

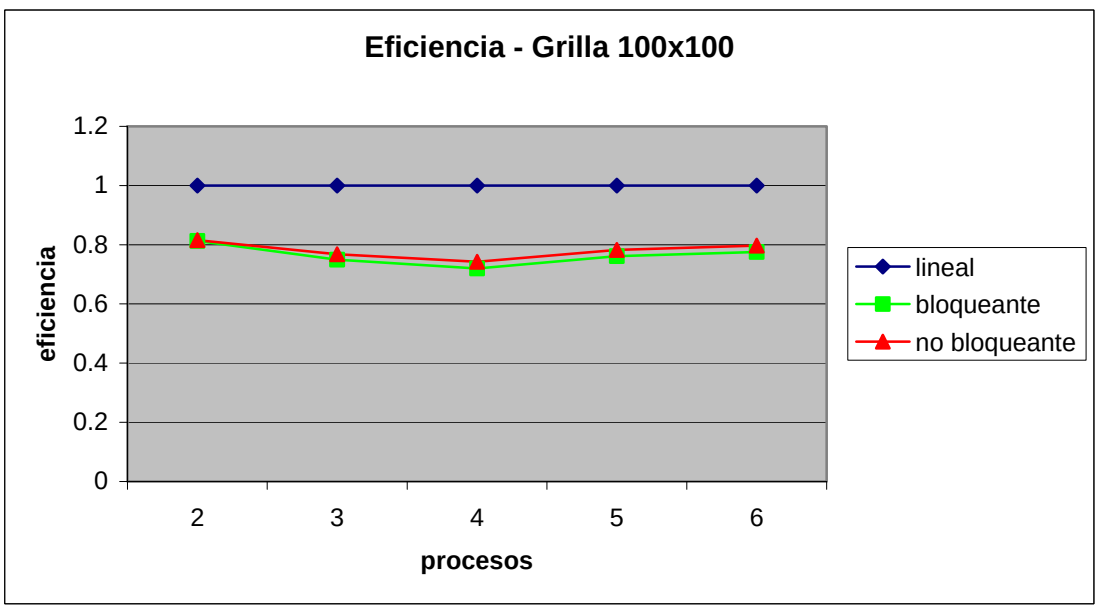

fig. 30. Eficiencia de la Grilla 100x100 en el cluster Speedy Gonzalez

Overhead

Como última medida de performance, se analiza la sobrecarga (overhead) de los algoritmos paralelos. La sobrecarga nos da una idea de cuánto trabajo extra ha realizado la versión paralela con respecto a su par serial.

Grilla ***n x n***	**Speedy Gonzalez**					
	Número de procesadores					
		2	**3**	**4**	**5**	**6**
50x50	**bloq**	3216	4044	2184	---	---
	no-bloq	2796	3030	696	---	---
100x100	**bloq**	16187	23485	27375	22002	20353
	no-bloq	15929	21265	24363	19567	17899

fig 31.a. Sobrecarga (en segs) obtenida en el cluster

Grilla ***n x n***	**Sócrates**	
	Número de procesadores	
		2
50x50	**bloq**	1845
	no-bloq	1845
100x100	**bloq**	8064
	no-bloq	8014

fig 31.b. Sobrecarga (en segs) obtenida en Sócrates

Al aumentar el tamaño de la grilla, vemos que hay un aumento notorio de la sobrecarga de los procesos paralelos. Esta sobrecarga se debe a varias razones: el balance de carga es más complejo cuando se aumenta el tamaño de la grilla, también aumentan los costos de comunicación y sincronización entre los procesos. En todas las pruebas realizadas se puede observar que hay una disminución de la sobrecarga cuando se utiliza la comunicación no bloqueante.

Capítulo 6

Conclusiones

En esta tesis se estudió y se desarrolló un simulador de reservorio de petróleo bidimensional de flujo bifásico *petróleo-agua*. Las ecuaciones del modelo matemático se derivaron utilizando la ecuación de continuidad, la ecuación de Darcy y las definiciones de compresibilidad para la roca y los fluidos asumiendo que la permeabilidad y la viscosidad son constantes. Se discretizó la grilla utilizando diferencias finitas equiespaciadas en ambas direcciones: longitudinal (x) y transversal (y). El simulador se implementó utilizando el método IMPES para la resolución del sistema de ecuaciones resultante.

Una vez obtenido el conjunto de ecuaciones que modelan el comportamiento del reservorio se procedió a analizar distintos métodos numéricos que resolvieran el primer paso del método IMPES. Se analizaron aquellos métodos iterativos que fueran a su vez candidatos a ser paralelizados. Se estudiaron varias alternativas. Los métodos directos no eran adecuados por las características de la matriz (rala) resultante, por lo que se procedió a estudiar algunos métodos iterativos. El método de Jacobi, el primer candidato a utilizar en este trabajo dado que las iteraciones son independientes entre sí, quedó descartado debido a que la cantidad de iteraciones necesarias para resolver el sistema de ecuaciones es proporcional al tamaño de la grilla que representa el reservorio a paralelizar. Por lo que son necesarias más iteraciones a medida que aumenta el tamaño de la grilla. Algo muy parecido sucedió con el método de Gauss-Seidel y no resultó útil porque tardaba mucho en converger.

Los otros métodos estudiados fueron el Block SOR y el SIP. El BSOR es el método que más comúnmente se utiliza para la resolución de este tipo de sistemas de ecuaciones. Si bien BSOR es el método con el que se obtiene mejor performance, es necesario estimar el factor de sobrerrelajación w óptimo para cada caso. Determinar dicho factor *a priori* utilizando el balance de carga ha demostrado ser muy costoso, lo cual lleva retrasos en términos de desarrollo del simulador. En cambio el SIP fue el método que mejor respondió en términos de tiempos de desarrollo y performance combinados. Estudiando las dependencias entre las variables en cada iteración, se determinó que la partición de dominio por ordenamiento *red-black* fue la que mejor resultados de independencia se obtenía.

Un tema a considerar en la programación paralela en cluster de computadoras es que la comunicación es tan importante como los cómputos. En las pruebas realizadas se pudo observar lo fundamental que es reducir los tiempos de comunicación y sincronización entre los procesos para lograr una buena performance de la implementación paralela. Un análisis

detallado de estos tiempos fue indispensable para realizar una adecuada optimización de los programas paralelos.

La partición del dominio resultó ser clave para poder obtener mejoras en la performance, especialmente cuando se trabajó en el cluster Speedy Gonzalez. El balanceo semi-dinámico de la carga introdujo un costo extra en las versiones paralelas, sin embargo, permitió un mejor aprovechamiento de los nodos heterogéneos del cluster y una reducción significativa en los tiempos de ejecución.

Para poder comparar las ventajas de solapar comunicación con cómputo (versión paralela no bloqueante) contra la versión que utiliza primitivas bloqueantes fue necesario realizar cambios significativos entre ambas versiones. Transformar una implementación con primitivas bloqueantes en una implementación con primitivas no bloqueantes no fue una tarea trivial. Se agregó la complejidad extra de verificar la disponibilidad de los datos al momento de ser utilizados.

Para obtener una buena performance de la solución paralela fue necesario minimizar los puntos de sincronización. Aquí se utilizaron varias técnicas: se redujo el número de veces que se verificaba la convergencia del método iterativo, se empleó un balance de carga semi-dinámico para que los procesos más veloces reduzcan sus tiempos ociosos y se utilizaron buffers en forma explícita para el envío de los mensajes (no teniendo que depender de los buffers que provee el sistema).

Como conclusión de este trabajo, se pudo observar que la computación paralela permitió atacar un problema de la vida real de grandes proporciones y resolverlo en menos tiempo. La programación paralela no es una tarea sencilla. Es difícil depurar un programa paralelo porque no se cuentan con muchas herramientas para tal fin. La utilización de primitivas no bloqueantes, si bien agrega una complejidad extra al momento de programar, logra mejoras en la performance que hace que valga la pena el esfuerzo.

Apéndice A

Cluster Beowulf Speedy Gonzalez

fig. 32 Speedy Gonzalez

Características del hardware

El cluster Speedy Gonzalez del Laboratorio de Sistemas Complejos del Departamento de Computación de la Facultad de Ciencias Exactas y Naturales es un cluster heterogéneo y está compuesto por 16 nodos con las siguientes características:

Hostname	CPU	MHz	Disco	RAM (Mb)	Habilitado
nodo1	Pentium III	733	ST3120022A	383	Si
nodo2	Pentium III	733	ST320423A	192	Si
nodo3	Pentium III	733	ST320423A	192	Si
nodo4	Pentium III	733	ST320423A	192	Si
nodo5	AMD Athlon	946	SV2042H	183	Si
nodo6	AMD Athlon XP 2000+	1666	ST320410A	256	Si
nodo7	AMD Athlon	946	SV2042H	183	Si
nodo8	AMD Athlon	946	SV2042H	183	Si

nodo9	AMD Athlon	946	FIREBALLlct20	183	Si
nodo10	AMD Athlon	946	FIREBALLlct20	183	Si
nodo11	AMD Athlon	1200	ST320423A	505	Si
nodo12	AMD Athlon	1200	ST320423A	505	Si
nodo13	AMD Athlon	1200	ST320410A	505	Si
nodo14	AMD Athlon	900	ST320410A	183	Si
nodo15	AMD Athlon	1200	ST320410A	505	Si
nodo16	AMD Athlon	900	ST320410A	183	Si

Características de los discos:

Marca	Modelo	Capacidad	Velocidad	Seek time	Interface
Segate	ST3120022A	120.9 Gb	7200 RPM	8.5 ms	Ultra ATA/100
Segate	ST320423A	20.49 Gb	5400 RPM	8.9 ms	Ultra ATA/66
Segate	ST320410A	20 Gb	5400 RPM	8.9 ms	Ultra ATA/100
Samsung	SV2042H	20.4 Gb	5400 RPM	9 ms	Ultra ATA/100
Quantum	FIREBALLlct20	20 Gb	5400 RPM	9 ms	Ultra ATA/100

Características del switch:

Marca	TRENDnet
Modelo	TEG-S240TX
Ports	24-port
Velocidad	10/100/1000 Mbps
Otras características	• auto-negociación y auto-MDIX en cada port • modos de transferencia Full/Half duplex para 10/100 Mbps • modo de transferencia Full duplex para 1000 Mbps • store-and-forward switching • 400 KBytes RAM para buffers de datos

Características de las placas de red:

Marca	3Com
Modelo	3C2000
Velocidad	10/100/1000 Mbps
Otras características	• auto-negociación transparente para que coincida con la velocidad del switch • TCP/UDP/IP checksums, reduce la carga al CPU

	• buffer de paquetes, Jumbo Frames • prioriza el tráfico y controla el multicast

Apéndice B

Nomenclatura

Símbolo	Definición	Dimensión
A_x	área transversal al flujo en la coordenada *x*	ft^2
A_y	área transversal al flujo en la coordenada *y*	ft^2
B_l	factor de volumen de la fase *l*	
ϕ	porosidad de la roca	
c_r	compresibilidad de la formación	psi^{-1}
k_{rl}	permeabilidad relativa a la fase *l*	
k	permeabilidad de la roca	
ρ	densidad	lbs/ft^3
μ_l	viscosidad de la fase *l*	cp
c_f	compresibilidad de un fluido	
$\mathrm{R_{so}}$	solubilidad del gas en el petróleo	
$\mathrm{R_{sw}}$	solubilidad del gas en el agua	
k_r	permeabilidad relativa	
P_{cow}	presión capilar agua-petróleo	psi
p_l	presión en la fase *l*	psi
S_l	saturación de la fase *l*	
k_x	permeabilidad en la coordenada *x*	
k_y	permeabilidad en la coordenada *y*	
β_c	factor de conversión, $(\beta_c = 1.127)$	
α_c	factor de ajuste, $(\alpha_c = 5.615)$.	
q_{osc}	caudal de extracción de petróleo	bbl/day
q_{wsc}	caudal de inyección de agua	bbl/day

$\partial Z / \partial x$	gradiente de profundidad de la coordenada *x*	
$\partial Z / \partial y$	gradiente de profundidad de la coordenada *y*	
Δx	distancia, en la coordenada *x*, entre dos puntos de la grilla	*ft*
Δy	distancia, en la coordenada *y*, entre dos puntos de la grilla	*ft*
T_{lx}	transmisibilidad direccional de la fase *l* en la coordenada *x*	
T_{ly}	transmisibilidad direccional de la fase *l* en la coordenada *y*	
V_b	volumen de una celda de la grilla	
Δt	cambio en un paso de tiempo	*days*

Bibliografía

[1] Al-Marhoon, N.O.; Carter, J. N.; Grattoni, C. A. and Dawe, R. A., *Effects of Permeability contrast in the Near Wellbore Flow Patterns-Comparison Between Experimental and Numerical Studies*, Society of Petroleum Engineers, 1999.

[2] Amdahl, G. M., *Validity of the single-processor approach to achieving large scale computing capabilities*, In AFIPS Conference Proceedings vol. 30 (Atlantic City, N.J., Apr. 18-20). AFIPS Press, Reston, Va., 1967, pp. 483-485.

[3] Amestoy, Patrick R.; Duff, Iain S.; L'Excellent, Jean-Yves and Li, Xiaoye S., *Impact of the Implementation of MPI point-to-point communications on the performance of two general sparse solvers*, CERFACS Technical Report TR/PA/03/14, France, 2003.

[4] Anderson, E. and Bai, Z. and Bischof, C. and Blackford, S. and Demmel, J. and Dongarra, J. and Du Croz, J. and Greenbaum, A. and Hammarling, S. and McKenney, A. and Sorensen, D., *Lapack User's Guide*, Third Edition, Society for Industrial And Applied Mathematics, Philadelphia, 1999.

[5] *Architecture and Programming of Parallel Computers:*
http://www.cs.nyu.edu/courses/fall98/G22.3033-003/lectures/lect2.pdf
http://www.cs.nyu.edu/courses/fall98/G22.3033-003/lectures/lect3.pdf
http://www.cs.nyu.edu/courses/fall98/G22.3033-003/lectures/lect9.pdf

[6] Aziz, K. and Settari, A., *Petroleum Reservoir Simulation*, Elsevier Applied Science Publishers, Great Britain, 1985.

[7] Blair, P. M. and Weinaug, C. F., *Solution of Two-Phase Flow Problems Using Implicit Difference Equations*, Esso Production Research Co., Houston, Texas.

[8] Brill, Stephen H. and Pinder, George F., *A Block Red-Black SOR Method for a Two-Dimensional Parabolic Equation Using Hermite Collocation*, University of Vermont Burlington, Vermont, U. S. A.

[9] Burden, Richard L. and Douglas Faires, J., *Análisis Numérico*, Grupo Editorial Americana, 1994.

[10] Chow, Edmond; Falgout, Robert D.; Hu, Jonathan J.; Tuminaro, Raymond S. and Yang, Ulrike Meier, *A Survey of Parallelization Techniques for Multigrid Solvers*.

[11] Cvetanovic, Zarka; Freedman, Edward G. and Nofsinger, Charles, *Efficient Decomposition and Performance of Parallel PDE, FFT, Monte Carlo Simulations; Simplex, and Sparse Solvers*, Digital Equipment Corporation Mid-Range Systems - Advanced Development.

[12] Douglas, Jim; Peaceman, D. W. and Rachford H. H., *A Method for Calculating Multi-Dimensional Immiscible Displacement*, Society of Petroleum Engineers, 1959.

[13] Duff, Iain S. and Van der Vorst, Henk A., *Developments and Trends in the Parallel Solution of Linear Systems*, CERFACS Technical Report TR/PA/99/10, France, 1999.

[14] Ertekin, T., *Basic Applied Reservoir Simulation*, Society of Petroleum, 2001.

[15] Foster, Ian, *Desinging and Building Parallel Programs*, Addison Wesley.

[16] Fox G. et al, *Solving Problems on Concurrent Processors*, Prentice Hall Inc, 1988.

[17] Huang, G. and Ongsakul, W., *Managing the Bottlenecks of a Parallel Gauss-Seidel Algorithm for Power Flow Analysis*, Dept. of Electrical Engineering.

[18] Kim, Jong Gyun, *Advanced Techniques For Oil Reservoir Simulation: Discrete Fracture Model And Parallel Implementation*, University of Utah, 1999.

[19] Kleppe, John, *Reservoir Simulation, Lecture Notes*, Sprint Semester 2003.

[20] Koester, D. P.; Ranka, S. and Fox, G. C., *A Parallel Gauss-Seidel Algorithm for Sparse Power System Matrices*; School of Computer and Information Science and The Northeast Parallel Architectures Center (NPAC), Syracuse University, Syracuse.

[21] LAPACK, Linear Algebra PACKage, http://www.netlib.org/lapack/

[22] Lemeire, Jan and Dirkx, Erik, *Causes of Blocking Overhead in Message-Passing Programs*, Parallel Systems lab, Vrije Universiteit Brussel, Pleinlaan 2, 1000 Brussels, Belgium.

[23] Levit, Creon, *Parallel Solution of Pentadiagonal Systems Using Generalized Odd-Even Elimination*, Proc. Supercomputing ’89, pp333-6, Reno, Nevada, IEEE Computer Society, 1989.

[24] Lippman, Stanley B. and Lajoie, Josée, *C++ Primer Third Edition*, Addison Wesley.

[25] Madabhushi, S. P. G.; Bellotti, D.; Clematis, A. and Fernades, P, *Parallelisation of a Finite Difference Code for Modelling Ground Water Flow*, Department of Engineering, University of Cambridge, Cambridge, CB2 1PZ England.

[26] Milano, Pablo, *Programación Paralela en un Sistema Beowulf Bajo Linux y MPI: Simulación Numérica Tridimensional de Problemas de Electrodeposición*, Tesis de Licenciatura en Cs. de la Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires.

[27] MPICH, http://www-unix.mcs.anl.gov/mpi/mpich/

[28] Odeh, A. S., *Comparison of Solutions to a Three-Dimensional Black-Oil Reservoir Simulation Problem*, Journal of Petroleum Technology, 33, 13-25, 1981.

[29] Pacheco, Peter S., *Parallel Programming with MPI*, Morgan Kaufmann Publishers, Inc., 1997.

[30] Press, William H.; Teukolsky, Saul A.; Vetterling, William T. y Flannery, Brain P., *Numerical Recipes in C – The Art of Scientific Computing*, Second Edition, Cambridge University Press.

[31] Regis, Sebastián P., *Desarrollo y aplicaciones de un simulador del flujo trifásico hacia un pozo de petróleo*, Tesis de Ingeniería en Química, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires.

[32] Reeve, J. S.; Merlin, Scurr and J. H., *Parallel Versions of Stone's Strongly Implicit Algorithm*, Department of Electronics and Computer Science, University of Southampton.

[33] Riyavong, S., *Experiments on Sparse Matrix Partitioning*, CERFACS Working Note WN/PA/03/32, France.

[34] Saad, Yousef, *Iterative Methods for Sparse Linear Systems*, Second Edition with corrections, 2000.

[35] Savioli, Gabriela B.; Morelli, M. de los Angeles; Regis, Sebastián P. y Bidner, M. Susana, *Análisis de estabilidad de un modelo semi-implícito de un pozo petrolífero*, Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería, Universitat Politècnica de Catalunya, España, Vol. 20, 1, 77-95, 2004.

[36] Silberchatz, Abraham and Galvin, Peter Baer, *Operating System Concepts*, Fifth Edition, Addison Wesley.

[37] Stone, Herbert L., *Iterative Solution of Implicit Approximations of Multidimensional Partial Differential Equations*, SIAM Journal on Numerical Analysis, Vol. 5, No. 3, 530-558, September 1968.

[38] Thomas, G. W., *Principles of Hydrocarbon Reservoir Simulation*, International Human Resources Development Corporation, Boston, USA, 1982.

[39] Todd, M. R.; O'Dell, P. M.; and Harasaki, G. J., *Methods for increased accuracy in numerical reservoir simulators*, Soc. Petr. Eng. J., 12, No. 6, 515, 1972.

[40] Van der Vorst, Henk A., *Lecture notes on iterative methods*, 1994.

[41] Xie, Dexuan, *New Block Parallel SOR Methods by Multi-Type Partitions*, Department of Mathematical Sciences, University of Wisconsin, Milwaukee, WI 53201-0413, USA.