UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA
INGENIERÍA EN EXPLORACIÓN Y EXPLOTACIÓN DE RECURSOS NATURALES – YACIMIENTOS

SIMULACIÓN NUMÉRICA DE YACIMIENTOS CON PROCESAMIENTO EN PARALELO

TESIS
QUE PARA OPTAR POR EL GRADO DE:
MAESTRO EN INGENIERÍA

PRESENTA:
OSCAR PEÑA CHAPARRO

TUTOR (ES) PRINCIPAL(ES)
DR. FERNANDO SAMANIEGO VERDUZCO, FACULTAD DE INGENIERÍA
DR. JORGE ALBERTO ARÉVALO VILLAGRÁN. FACULTAD DE INGENIERÍA
DR. VÍCTOR HUGO ARANA ORTÍZ, FACULTAD DE INGENIERÍA
DR. RODOLFO GABRIEL CAMACHO VELÁZQUEZ, PEMEX
DR. JOSÉ ANTONIO GONZÁLEZ GUEVARA, PEMEX

MÉXICO, D. F. SEPTIEMBRE 2016

JURADO ASIGNADO:

Presidente:	DR. FERNANDO SAMANIEGO VERDUZCO
Secretario:	DR. JORGE ALBERTO ARÉVALO VILLAGRÁN
Vocal:	DR. VÍCTOR HUGO ARANA ORTÍZ
1er Suplente:	DR. RODOLFO GABRIEL CAMACHO VELÁZQUEZ
2do Suplente:	DR. JOSÉ ANTONIO GONZÁLEZ GUEVARA

Lugar o lugares donde se realizó la tesis: México, Distrito Federal

TUTOR DE TESIS:

Dr. Víctor Hugo Arana Ortíz

A mis padres Gloria y Ángel,
por su amor incondicional,
por su firmeza y disciplina,
por su apoyo y,
por estar siempre.

Gracias.

Oscar

El día de hoy es una fecha importante para mí, donde no sólo concluyo un trabajo de tesis, sino un ciclo que me parecía en demasía largo; durante este tiempo he tenido la fortuna de conocer nuevos amigos y reencontrarme con otros quiénes a pesar del tiempo y distancia, mantienen vivos los recuerdos de años pasados, demostrándome que la verdadera amistad perdura más allá de los obstáculos que puedan presentarse, propios de nuestro crecimiento y desarrollo. En todo momento me han demostrado su apoyo y otorgado palabras de aliento para la conclusión de este trabajo, a ustedes dedico la conclusión de este material y ante todo agradezco su amistad.

También agradezco a la máxima casa de estudios, la Universidad Nacional Autónoma de México (UNAM), institución de la cual formo parte oficialmente desde hace muchos años y que me ha formado en sus espacios desde la Escuela Nacional Preparatoria hasta la División de Estudios de Posgrado, sin olvidar mencionar la Facultad de Ingeniería, lugares que me han brindado su conocimiento y visión del mundo, permitiéndome formar un criterio para afrontar los retos cotidianos y fijar metas en la vida.

Agradezco a los sinodales su tiempo para la revisión de este trabajo, así como su amistad y consejos a lo largo de este camino, Dr. Fernando Samaniego Verduzco, Dr. Rodolfo G. Camacho Velázquez, Dr. José Antonio González Guevara, Dr. Jorge A. Arévalo Villagrán y en especial al Dr. Víctor Hugo Arana Ortiz, quien no cesó en promover el crecimiento profesional de sus alumnos.

Finalmente, agradezco a mi familia, mis hermanos Ángel, José Luis y Juan Carlos su apoyo durante estos años y a mis padres Ángel y Gloria quiénes a pesar de las adversidades siempre dieron todo por su familia.

"Gracias Totales"

OPC

RESUMEN

En este trabajo se presenta el desarrollo de un simulador numérico de yacimientos con el modelo de aceite negro, empleando técnicas de programación en paralelo. Para esto, se realiza una revisión de la bibliografía existente y se hace referencia a los trabajos más relevantes tanto de ingeniería petrolera, como de cómputo científico; posteriormente se continúa con el proceso de desarrollo del simulador. El lenguaje de cómputo usado es FORTRAN 90 y el modelo de programación en paralelo es de paso de mensajes, para el cual se emplean las librerías de MPI (*Message Passing Interface*), para establecer la comunicación entre los procesos que así lo requieran.

Para la programación en paralelo se presentan un conjunto de elementos con la finalidad de dar un panorama general del cómputo de alto rendimiento; en ellos se abordan diferentes temas como la arquitectura de computadoras paralelas con sus tipos de conexión, los modelos y paradigmas de programación existentes mayormente usados, la metodología teórica que se sigue para el diseño de algoritmos de este tipo de aplicaciones, y las características de las librerías de paso de mensajes.

Este simulador de aceite negro se desarrolló a partir de la solución numérica del modelo matemático por medio del método de diferencias finitas bajo una formulación totalmente implícita, se considera flujo multifásico (aceite, gas y agua) en tres dimensiones y de forma isotérmica, además, se maneja doble porosidad (aplicable a yacimientos naturalmente fracturados) a través de una función de transferencia de fluidos matriz fractura con una permeabilidad, propuesta por Kazemi. El modelo de pozo empleado es el de Peaceman con el cual se puede producir a gasto o presión constante.

Los estudios de simulación requieren de una gran cantidad de información y de procesamiento, para obtener buenos resultados se realizan simulaciones de campo a gran escala que no pueden ejecutarse en computadoras seriales por la alta demanda de recursos de memoria y procesador, y en el caso de satisfacer estos requerimientos, el tiempo de ejecución es considerablemente alto. Para resolver este problema aplicando las técnicas de programación en paralelo, se empleó la descomposición de dominio para seccionarlo, asignar una parte a cada uno de los procesadores, y combinar todas las soluciones parciales obtenidas, que permitan conseguir un resultado final total; esta técnica de descomposición de dominio se usó como una solución de ingeniería para problemas de gran escala.

En este trabajo se presenta el diseño del algoritmo paralelo, su estructura y los criterios considerados. El simulador se validó con software comercial y la versión serial del mismo. Fundamentalmente se analizaron dos aspectos del programa: su desempeño para simulaciones a gran escala y el comportamiento de la aplicación en paralelo; para esto, en los casos de aplicación se realizaron corridas a gran escala, con un número diferente de procesadores para observar su comportamiento, rendimiento y tiempos de ejecución, por otro lado, se emplearon algunas métricas de cómputo paralelo para evaluar el simulador y demostrar los beneficios obtenidos, reportando los resultados con su análisis respectivo y conclusiones.

LISTA DE FIGURAS

Pág.

FIG. II.1 Idealización de un yacimiento naturalmente fracturado ……... 8
FIG. II.2 Representación de la interacción de los sistemas de fracturas (*f*) y matriz (*m*) ……. 9
FIG. II.3 Sistema matricial considerando un pozo con presión especificada …….. 29
FIG. II.4 Sistema matricial considerando un pozo con gasto especificado ……. 31

FIG. III.1 Clasificación de arquitecturas de computadoras paralelas …….. 40
FIG. III.2 Sistema de memoria compartida (Shared Memory) …… 41
FIG. III.3 Sistema de memoria distribuida (Distributed Memory) …….. 42
FIG. III.4 Switch de conexión cruzado …… 43
FIG. III.5 Arreglo de conexión de malla …….. 44
FIG. III.6 Conexión Hipercubo …… 45
FIG. III.7 Conexión de cluster ……. 45
FIG. III.8 Estructura maestro esclavo …….. 49
FIG. III.9 Estructura básica de un programa SPMD …… 49
FIG. III.10 Estructura de línea de datos ……... 50
FIG. III.11 Paradigma dividir y conquistar como árbol virtual ……... 51
FIG. III.12 Etapas de desarrollo de algoritmos paralelos …… 53

FIG. IV.1 Curvas de permeabilidad relativa agua-aceite para la fractura y matriz ….. 63
FIG. IV.2 Curvas de presión capilar para la fractura y matriz…….. 64
FIG. IV.3 Configuración de dominios para el método de Schwarz alternado ……. 72
FIG. IV.4 Discretización en subdominios de un dominio global ……. 76
FIG. IV.5 Discretización en subdominios considerando dos colores ……..... 76
FIG. IV.6 Diagrama de flujo del simulador numérico de yacimientos en paralelo …. 80

FIG. V.1 Comportamiento de p_{wf} y *RGA* para el Pozo 1 en la versión serial y en paralelo……... 87
FIG. V.2 Comportamiento de Presión en el modelo para t=1 día (izquierda), t=2000 días (derecha)……. 87
FIG V.3 Aceleración del simulador en paralelo…… 88
FIG V.4 Comportamiento de Presión de fondo fluyendo en el Pozo 1 y comparación con versión serial y eclipse …… 90
FIG V.5 Comportamiento de Presión en el modelo para t=400 días ……... 91

LISTA DE TABLAS

Pág.

Tabla V.1 Datos de entrada……...…..……... 84
Tabla V.2a Información PVT aceite……..……..…….. 85
Tabla V.2b Información PVT gas……..……..... 85
Tabla V.2c Información PVT agua..…….…..……... 85
Tabla V.3a Permeabilidad relativa gas-aceite……..……..…..……... 86
Tabla V.3b Permeabilidad relativa agua-aceite….……..……..…..……... 86
Tabla V.4 Resultados de procesamiento para diferente número de procesadores…… 88
Tabla V.5 Datos de entrada……... 89
Tabla V.6a Permeabilidad relativa gas aceite……... 89
Tabla V.6b Permeabilidad relativa agua aceite……. 90

ÍNDICE **Pág.**

RESUMEN…..……..……..... ***i***

LISTA DE FIGURAS…….…... ***ii***

LISTA DE TABLAS…….…….... ***iii***

ÍNDICE…..…….. ***iv***

CAPÍTULO I. INTRODUCCIÓN…..……. **1**

I.1. Revisión Bibliográfica…….……... 2
I.2. Organización…..…….. 5

CAPÍTULO II. SIMULACIÓN NUMÉRICA DE YACIMIENTOS….…… **7**

II.1. Simulación de Yacimientos Naturalmente Fracturados ……... 7
II.2. Formulación Matemática de las Ecuaciones de Flujo ……..……... 9
II.2.1. Ecuaciones para las Fracturas ……..... 9
II.2.2. Ecuaciones para la Matriz ……... 10
II.2.3. Función de Transferencia de Fluidos Matriz-Fractura ……... 11
II.2.4. Condiciones Iniciales y de Frontera …… 12
II.3. Formulación Numérica de las Ecuaciones de Flujo ……..... 13
II.3.1. Ecuaciones de las Fracturas en Diferencias Finitas …… 14
II.3.2. Ecuaciones de los Bloques de Matriz en Diferencias Finitas ……..... 17
II.4. Solución del Sistema de Ecuaciones ……. 18
II.4.1. Linealización del Sistema de Ecuaciones ……... 19
II.4.2. Reducción Matricial del Sistema de Ecuaciones …… 23
II.5. Tratamiento de Pozos en Simulación Numérica de Yacimientos …… 25
II.5.1. Modelo del Pozo …….. 26
II.5.2. Solución del Modelo del Pozo ……..... 28
II.5.2.1. Presión Especificada del pozo ……..... 29
II.5.2.2. Gasto Especificado del pozo ……... 30

CAPÍTULO III. PROGRAMACION EN PARALELO …….. **32**

III.1. Conceptos de Cómputo en Paralelo ……..... 32
III.1.1. Conceptos Fundamentales …….. 33
III.1.2. Elementos de Diseño …….. 34
III.2. Arquitecturas de Computadoras Paralelas ……... 39
III.2.1. Clasificación ……... 39
III.2.2. Arquitecturas Paralelas ……... 41
III.2.3. Esquemas de Comunicación ……... 43
III.3. Modelos de Programación en Paralelo ……..... 46
III.4. Paradigmas de Programación en Paralelo …… 48
III.5. Diseño de Programas en Paralelo ……..... 52

III.5.1. Partición …….. 53
III.5.2. Comunicación e Interferencia de Tareas ……..... 54
III.5.3. Aglomeración …….. 54
III.5.4. Mapeo ……. 55
III.6. MPI (Message Passing Interface) ……..... 56
III.6.1. Especificaciones de MPI ……..... 57
III.6.1.1. Comunicaciones Punto a Punto ……. 57
III.6.1.2. Comunicaciones Colectivas ……... 58
III.6.1.3. Comunicadores …….. 59
III.6.1.4. Grupos y Topologías de Procesos …….. 60
III.6.1.5. Vínculos para C y Fortran e Interfaz de Perfilamiento …….. 60
III.6.2. Tipos Derivados de Datos ……... 61

CAPÍTULO IV. SIMULADOR DE YACIMIENTOS EN PARALELO ……... 62

IV.1. Consideraciones de Simulación de Yacimientos Naturalmente Fracturados .. 63
IV.2. Diseño del Simulador en Paralelo …… 65
IV.2.1. Proceso de Simulación …… 65
IV.2.2. Estrategias de Paralelización …….. 67
IV.2.2.1. Modelo de Programación ……..... 68
IV.2.2.2. Paradigma de Programación …… 68
IV.2.3. Partición, Comunicación, Aglomeración y Mapeo ……..... 69
IV.2.4. Algoritmo Paralelo …….. 70
IV.3. Descomposición de Dominio ……... 71
IV.3.1. Método Alternado de Schwarz ……... 72
IV.3.2. Método de Schwarz discretizado en mallas coincidentes 73
IV.3.3. Múltiples dominios ……..... 75
IV.3.4. Aspectos relevantes de métodos de descomposición de dominios …..... 77
IV.3.5. Métricas de programas en paralelo ……..... 78
IV.4. Diagrama de flujo del Simulador ……..... 79

CAPÍTULO V. CASOS DE APLICACIÓN……..... 84

V.1. Modelo de simulación de aceite negro en un medio homogéneo ……..... 84
V.2. Modelo de simulación de aceite negro en un medio de doble porosidad …..... 88

CAPÍTULO VI. CONCLUSIONES Y RECOMENDACIONES ……... 92

NOMENCLATURA ……. **94**

BIBLIOGRAFÍA …….. **97**

APÉNDICE……..……... **102**

A. ECUACIONES DE FLUJO MULTIFÁSICO EN MEDIOS POROSOS…… **102**

I. INTRODUCCIÓN

El cómputo de alto rendimiento se ha empleado generalmente para realizar cálculos a gran escala en paralelo dentro de áreas estratégicas del gobierno, meteorología, tecnología aérea y/o espacial, etc.; en general, se emplean para resolver una variedad amplia de problemas de interés comercial y científico, en los cuales, los modelos matemáticos que describen los procesos físicos y químicos son esenciales. **Golub y Ortega (1997)**, establecen que el supercómputo tiene un gran impacto en la ciencia y en la ingeniería; los más recientes y usos extensos de las computadoras se realizan en estas dos áreas y más específicamente, se emplean para obtener soluciones de modelos matemáticos que representen situaciones físicas. El modelado en computadoras puede ahorrar millones de dólares en comparación a desarrollar prototipos y forma parte del cómputo científico.

En el cómputo científico, el procesamiento en paralelo ha tenido un gran impacto en disciplinas como ingeniería, ciencias, negocios y medicina, donde se requiere de velocidades de procesamiento que difícilmente pueden alcanzar las computadoras convencionales, ya que involucran el manejo de grandes cantidades de datos y la realización de un buen número de iteraciones. El procesamiento en paralelo, involucra el uso de varios elementos como arquitecturas paralelas, algoritmos paralelos, lenguajes de programación y análisis de comportamiento de programas paralelos, que permiten resolver problemas complicados, empleando varias computadoras o procesadores que trabajan de forma simultánea cooperando entre ellos para obtener un resultado; el cómputo en paralelo permite fraccionar el problema (ya sea de forma geométrica en sus cálculos o en los datos) y cada una de las partes asignarla a cada uno de los procesadores con que disponemos.

Las computadoras paralelas con varias unidades centrales de proceso han existido desde hace varias décadas; sin embargo, hasta ahora se les ha encontrado una aplicación realmente significativa. Existen varios tipos de máquinas paralelas y se diferencian entre ellas por la forma en que el procesador maneja las instrucciones y los datos.

Una de las aplicaciones del cómputo en paralelo se encuentra dentro de la ingeniería petrolera en la simulación numérica de yacimientos. Esta es un área donde se estudia el comportamiento de los fluidos al desplazarse en los poros interconectados de un yacimiento bajo diferentes esquemas de explotación, permitiéndonos optimizar su producción para obtener la máxima recuperación económica, y técnicamente posible; en la simulación de yacimientos se modela el flujo y cambio de fase de los fluidos dentro del medio poroso, siendo la clave el poder resolver en cortos tiempos grandes sistemas de ecuaciones obtenidos a partir de modelos numéricos derivados de modelos matemáticos representados por medio de ecuaciones diferenciales. Estos estudios se hacen por medio de simuladores que pueden ser de carácter composicional, gas seco, aceite negro, etc., y que permiten simular procesos que van desde el depresionamiento natural (producción primaria) hasta recuperación mejorada con procesos térmicos, químicos, entre otros. Mediante el uso de un simulador se puede explotar un yacimiento muchas veces de forma virtual para obtener el mejor programa de explotación ya que en la realidad los yacimientos sólo pueden explotarse una vez.

A través de los años, los simuladores han adquirido gran importancia y su desarrollo ha ido de la mano con los avances en el cómputo científico, permitiendo realizar gran cantidad de procesamiento de datos y manejar una amplia variedad de información de diversos volúmenes, además, los avances en áreas como análisis numérico, matemáticas, optimización de algoritmos, ciencias de la computación, entre otras, han contribuido con su crecimiento y alcance. Sin embargo, el reto continúa y actualmente se busca el poder crear modelos de simulación cada vez más robustos en los que se pueda plasmar con mayor aproximación el problema físico, situación que significa incluir las heterogeneidades presentes en el yacimiento y elementos que afectan el flujo, entre las que encontramos la presencia de acuíferos, fracturas, fallas, discontinuidades o discordancias, cambios litológicos y de facies, variación de las propiedades petrofísicas con respecto al tiempo, etc. Los simuladores de yacimientos se han convertido en una herramienta fundamental para el desarrollo de campos, en un principio, la capacidad de cómputo (en cuanto a operaciones realizadas y memoria disponible) era limitada por lo que los modelos de simulación se construían con poca información, sin embargo, el desarrollo del equipo de cómputo ha tenido un gran avance en estos aspectos. Actualmente existen varios trabajos basados en la paralelización de simuladores numéricos de yacimientos de los que presentamos algunos de los más importantes en la siguiente sección.

El objetivo de este trabajo es el de construir un simulador numérico de yacimientos en paralelo empleando el modelo de aceite negro, permitiéndonos representar el flujo en el medio poroso de manera apropiada a escala de campo, con suficiente resolución en espacio de tal manera que se pueda hacer una descripción adecuada de las heterogeneidades del yacimiento y ejecutándose a una mayor velocidad de procesamiento. El modelo numérico de este simulador será desarrollado en diferencias finitas con una formulación totalmente implícita, además, incluirá flujo multifásico bajo condiciones isotérmicas en tres dimensiones, doble porosidad y una permeabilidad.

La combinación del cómputo científico y la ingeniería petrolera ha dado grandes frutos al proveer de herramientas de trabajo de alto rendimiento cuya labor no podía ser realizada de forma manual por los ingenieros, estas herramientas han contribuido al desarrollo y administración de la industria al brindar de elementos al ingeniero para la toma de decisiones, por lo que es aconsejable que además de la simulación de yacimientos estás áreas se utilicen para la solución de otro tipo de problemas.

I.1. Revisión Bibliográfica

El cómputo en paralelo se ha empleado en una gran cantidad de problemas que requieren el manejo de mucha información y de procesamiento en áreas como la meteorología, astronomía, aeronáutica, etc.; la simulación de yacimientos no ha sido la excepción por lo que existen varios trabajos que se han tomado como base, a continuación se mencionan algunos de los más relevantes, comenzando por el artículo presentado por **Chien y cols. (1987)** en donde se observó el uso de la vectorización y procesamiento en paralelo en un simulador numérico de yacimientos multipropósito que incluye el modelo de aceite negro, composicional y procesos de inyección de vapor, se obtuvo un aumento significativo en la velocidad de solución de problemas diferentes cuyo valor dependía del número de procesadores usados y del tamaño del problema; dentro de la simulación de

yacimientos cabe resaltar que previamente a la paralelización se realizaron varios esfuerzos por programar simuladores en máquinas de procesamiento vectorial alcanzando buenos resultados, **Young y Zarantonello (1991)** mostraron un alto desempeño de simulación bajo procesamiento vectorial al resolver problemas muy grandes en tiempos prácticos con los modelos de aceite negro y composicional, el mayor problema fue de un millón de celdas para veinte años de simulación. Posteriormente al procesamiento vectorial, el esfuerzo de simulación se centró en el procesamiento paralelo.

Un primer trabajo para comenzar a aprender sobre éste tema fué el realizado por **Killough (1993)**, quien presentó el estado del arte en ese momento de la tecnología de cómputo en paralelo desde conceptos generales como arquitectura paralela, balance de carga, etc., hasta su utilidad en la rutina de solución de sistemas de ecuaciones así como para la construcción de dicho sistema, este trabajo proporciona una idea clara del procesamiento en paralelo y su interrelación con la ingeniería petrolera.

En algunos trabajos se ha puesto énfasis en partes del proceso de simulación, las cuales se consideran críticas, por lo que requieren el mayor esfuerzo de paralelización, tal es el caso de **Barua y Horne (1989)** quienes mostraron que la combinación de técnicas aplicadas al problema lineal y no lineal, contribuyen en la aceleración total de los simuladores; el trabajo desarrollado por estos autores se centró en la solución del jacobiano y en la implementación del método Quasi-Newton, para la linealización del sistema de ecuaciones, mejorando el comportamiento del simulador en su versión serial y en paralelo; **Anguille y Killough (1995)** presentaron su trabajo donde se centraron en las estrategias de balance de carga dinámica y estática, para evitar el exceso de comunicación entre procesos, reducir de esta manera la latencia o tiempo de espera de información para procesamiento y extender el ancho de banda, se optimizó el algoritmo y se mejoró la eficiencia del programa paralelo. Otros autores han llevado a la paralelización más allá de la solución del sistema de ecuaciones, como **Zhuang y Zhu (1995),** quienes realizaron la paralelización de un simulador de yacimientos de aceite negro bajo la formulación IMPES, empleando diferentes librerías de comunicación, MPI y NX de Intel, como herramientas para la programación en paralelo y observaron comportamientos similares en ambas versiones.

Algunos autores han realizado modificaciones de simuladores existentes como **Hemanth y Young (1996),** quienes presentaron la adaptación de un simulador de yacimientos para cómputo en paralelo tanto para aceite negro como composicional, originalmente el simulador se diseñó para procesamiento con vectores y se adaptó para procesamiento en paralelo en máquinas con memoria compartida, mostrando un mejor rendimiento.

Otros trabajos han realizado estudios en diferentes tipos de máquinas paralelas; tal es el caso de **Zhiyuan y cols. (1995),** que realizaron trabajos de simulación de yacimientos empleando el modelo de aceite negro en computadoras paralelas con memoria distribuida y en clusters; los estudios se basaron en los métodos de solución del sistema de ecuaciones; **Silva y cols. (2003)** presentaron la paralelización de un simulador de aceite negro en máquinas SMP (*symmetric multiprocessing*), se basaron en su estudio en el simulador BOAST, simulador multifásico, tridimensional de aceite negro y observaron que, la mayor parte de tiempo de cómputo, se consumía durante la formación del jacobiano y la solución

del sistema de ecuaciones donde se emplearon las técnicas de programación en paralelo y el paso de mensajes con MPI demostrando una mayor eficiencia.

En la simulación de yacimientos se ha destacado el hecho de poder representar el comportamiento de flujo empleando un gran número de celdas, con la finalidad de tener una representación mejor del movimiento de las fases en el yacimiento, por lo que un aspecto importante de un simulador es el poder manejar una alta cantidad de bloques, **Kaarstad y cols. (1995)** presentaron un simulador de yacimientos con flujo de dos fases, una formulación totalmente implícita y en tres dimensiones para grandes simulaciones (masivas) implementando descomposición de dominio, concluyeron a partir de experimentos, que el simulador era capaz de realizar corridas de cientos o miles de celdas en pocos minutos; **Zhang y cols. (2001)** destacó una serie de técnicas de cómputo en paralelo para simulaciones de gran escala, implementadas en un simulador comercial de flujo multifásico y multicomponente, obteniendo mejoras en cuanto a capacidad de número de celdas y eficiencia, **Dogru y cols. (2002)** presentaron un simulador en paralelo para yacimientos a gran escala, empleando el modelo de aceite negro e integrándolo a un ambiente gráfico para el pre- y post- proceso, se paralelizó en el *solver* lineal, así como en el refinamiento local y manejo de pozos. Finalmente un informe importante fue el de **Yang y cols. (2005)** quienes desarrollaron un trabajo enfocado a simulaciones de gran escala, en el que lograron simular cerca de 10 millones de celdas de un yacimiento y establecieron, como clave de la simulación numérica de yacimientos la solución del sistema de ecuaciones, así como, la implementación de algoritmos eficientes en paralelo empleados para el cálculo de los coeficientes y de las saturaciones de las celdas.

Estos trabajos han sido importantes en la simulación numérica de yacimientos y pertenecen a épocas diferentes, se consideran un buen comienzo para entrar a esta disciplina. Por otro lado, en el área del cómputo científico encontramos otro tipo de desarrollos donde se aplicaron las técnicas de procesamiento en paralelo que nos ayudarían a entender de forma integral este problema, mismos que se mencionan a continuación.

Los trabajos siguientes, si bien no se relacionan con la simulación de yacimientos y ni siquiera con la ingeniería petrolera, si lo hacen con la programación en paralelo, y permitieron la realización de este trabajo, como lo es el desarrollado por **Pancake (1996),** quien planteó una serie de ideas y cuestionamientos para decidir si es conveniente, o no, paralelizar la solución computacional de un problema, brindando además, un panorama general de esta disciplina; **Golub y Ortega (1997)** presentan las bases del cómputo científico y dan una introducción en el procesamiento paralelo de una forma clara; **Cismaşiu (2002)** presenta la aplicación de esta técnica en un problema de ingeniería civil; **Foster (1995)**, **Roosta (1999)** y **Barney (2007),** muestran en sus publicaciones las bases del cómputo paralelo, las características del hardware con sus ventajas y desventajas, las metodologías de diseño de algoritmos y casos de aplicación de acuerdo a la arquitectura de los sistemas de cómputo; **Buyya (1999)** presenta una serie de conceptos relacionados a los modelos de programación en paralelo y a los paradigmas existentes; y finalmente encontramos las publicaciones de **Alonso (1997)** y **Dongarra y cols. (1995)** quienes presentan un panorama general de las librerías de MPI, las cuales son una de las herramientas empleadas para el desarrollo de este tipo de aplicaciones.

I.2. Organización

El presente trabajo está compuesto de seis capítulos y un apéndice, en los que se describe el esfuerzo invertido para su realización, con sus resultados y conclusiones, los cuales permitieron alcanzar el objetivo planteado. El primer capítulo incluye una introducción del cómputo científico y su relación con la ingeniería petrolera, específicamente en el área de simulación numérica de yacimientos. También, se hace referencia a una serie de trabajos que se han realizado obteniendo resultados relevantes, mostrando los beneficios que se pueden llegar a tener al combinar técnicas computacionales con problemas físicos.

En el segundo capítulo se presenta la descripción del problema a resolver, comenzando con la formulación matemática del problema de flujo de fluidos en medios porosos naturalmente fracturados en tres dimensiones, y que deriva en el desarrollo de las ecuaciones de flujo de cada una de las fases presentes en el yacimiento; también se muestra el desarrollo numérico de los modelos matemáticos para resolver las ecuaciones, así como su linealización por el método de Newton-Raphson, a través del cual se genera el sistema matricial de ecuaciones que se resuelve en el simulador, por otro lado se enseña el modelo de pozo que se emplea para el cálculo de la presión de fondo fluyendo o los gastos según sea el caso.

Dentro del tercer capítulo se presentan los conceptos básicos de la programación en paralelo que nos permiten adentrarnos en esta área del cómputo científico, se muestra de forma general parte de la arquitectura de computadoras paralelas con algunos tipos de ellas y las formas posibles de interconexión de los procesadores, adicionalmente, se dan los modelos y paradigmas existentes de programación en paralelo que nos permitirán tomar una decisión con respecto al camino que deberá seguir el desarrollo del programa de acuerdo a los recursos disponibles y a nuestras necesidades, posteriormente, se enseñan los elementos que se deben de considerar para el diseño de algoritmos, así como los pasos teóricos que se recomiendan seguir, finalmente, se realiza una descripción de las librerías de paso de mensajes (MPI), mismas que nos sirven para comunicar información entre diferentes procesos y que fueron utilizadas como herramienta de desarrollo de la aplicación en paralelo.

El cuarto capítulo está dedicado al simulador en paralelo y en él se establece el proceso de simulación, los esquemas y consideraciones adoptados para su programación, la estrategia de paralelización y el algoritmo. Por otro lado, se brinda una explicación teórica del método de descomposición de dominio empleado, derivado de los métodos de Schwarz, y se establecen las técnicas de implementación, lo anterior se debe a que no es un método que pueda ser paralelizado directa o naturalmente.

En el quinto capítulo se presentan los resultados obtenidos con el simulador en paralelo para diferentes casos de aplicación, comenzando con su validación tanto con software comercial como con su versión serial, y posteriormente, se presenta un caso donde se evalúa el rendimiento del algoritmo para diferente número de procesadores y su comparación con respecto a la versión en serie.

En el capítulo sexto, se presentan las conclusiones referentes al procesamiento en paralelo y la simulación numérica de yacimientos, así como sus áreas de oportunidad. Finalmente, en el apéndice se presenta material de apoyo para la comprensión y desarrollo de este trabajo, así como el desarrollo de las ecuaciones de flujo multifásico en medios porosos.

II. SIMULACIÓN NUMÉRICA DE YACIMIENTOS

La simulación numérica de yacimientos se utiliza para resolver problemas que no pueden resolverse por otro medio. Constituye la única manera para describir de forma cuantitativa el flujo de fases múltiples a través de un medio heterogéneo, obteniendo la producción en función de las propiedades del yacimiento, la demanda en el mercado, las estrategias de inversión y las regulaciones gubernamentales, **Mattax y Dalton (1990)**. La simulación implica la construcción y operación de modelos cuyo comportamiento coincide con el que presenta el yacimiento; estos modelos pueden ser físicos o matemáticos; un modelo matemático es un conjunto de ecuaciones que, de acuerdo a ciertas suposiciones, describe el proceso físico activo en el yacimiento siendo más barato que los modelos físicos, **Coats (1969)**, estas ecuaciones se encuentran expresadas en derivadas parciales y representan conservación de masa y/o energía.

El objetivo de la simulación numérica de yacimientos es proporcionar al ingeniero de diseño de explotación, herramientas confiables para predecir el comportamiento de los yacimientos de hidrocarburos con diferentes condiciones o esquemas de operación; estas herramientas son los simuladores numéricos de yacimientos y surgen de la combinación de física, matemáticas, ingeniería de yacimientos, ingeniería de computación, etc.

El uso de los simuladores permite reducir el riesgo asociado a la elección del plan de explotación y por lo tanto minimiza los flujos de efectivo negativos. Los factores que contribuyen al riesgo son: la complejidad del yacimiento reflejada en la anisotropía de las propiedades de la roca, variaciones regionales de las propiedades de los fluidos y permeabilidades relativas, los mecanismos de recuperación, etc., **Ertekin, Abou-Kassem, y King (2001)**. Actualmente, la confiabilidad de los simuladores modernos y la disponibilidad de computadoras de alto rendimiento, vuelven a la simulación una tarea práctica para la toma diaria de decisiones en los yacimientos de todos los tamaños.

En este capítulo se presenta el desarrollo de un simulador de yacimientos naturalmente fracturados, empleando el modelo de aceite negro y el concepto de doble porosidad de **Warren y Root (1963)**; se muestra la obtención del modelo matemático como un conjunto de ecuaciones en derivadas parciales y se especifican las condiciones de frontera e iniciales; se continúa con la solución numérica del modelo matemático empleando el método de diferencias finitas de donde se obtiene un sistema de ecuaciones, el cual para resolverlo se linealiza por el método de Newton Raphson; finalmente, se muestra el modelo del pozo empleado en el simulador, el cual se maneja de forma totalmente implícita para tener mayor estabilidad durante las corridas de simulación.

II.1. Simulación de Yacimientos Naturalmente Fracturados

Las técnicas convencionales de simulación de yacimientos han demostrado ser inadecuadas cuando se aplican directamente al estudio de sistemas fracturados. Estos sistemas se caracterizan por valores extremos de propiedades de roca y fluidos, como lo son las grandes discontinuidades que existen en porosidad, permeabilidad y saturación en el yacimiento; además, la mayoría de los fluidos se encuentran almacenados en bloques de matriz de muy baja permeabilidad y de diferentes tamaños, mientras que la movilidad del

fluido se realiza en las fracturas interconectadas que representan un volumen pequeño de hidrocarburos, pero con muy alta permeabilidad, **Rossen (1977)**. Debido a lo anterior, un yacimiento naturalmente fracturado puede conceptualizarse formado por un par de sistemas superpuestos, uno continuo y otro discontinuo. El primero de ellos está formado por una red de fracturas, a través de la cual se realiza el flujo de fluidos de la formación hacia los pozos; es un sistema de alta permeabilidad y baja porosidad efectiva. El segundo sistema es el de los bloques de matriz que contienen la mayor parte del volumen poroso y sirven como fuentes o sumideros de las fracturas. Cada sistema tiene propiedades petrofísicas diferentes y la interacción entre los bloques de matriz y el sistema de fracturas definirán en gran medida la recuperación de los hidrocarburos contenidos en el yacimiento. La interacción entre los dos sistemas se representa por medio de una función de transferencia de fluidos matriz/fractura, que se basa en una extensión de la ecuación desarrollada por **Warren y Root (1963)** y toma en cuenta a la presión capilar, la fuerza de gravedad y a las fuerzas viscosas.

La idealización mencionada de considerar a las fracturas como un medio continuo puede aplicarse a muchos sistemas heterogéneos en los que constituyen la vía principal de flujo para producción o inyección, mientras que los bloques de matriz por su alta capacidad de almacenamiento, alimentan al medio continuo. En la **Fig. II.1** se muestra de forma esquemática la idealización propuesta por **Warren y Root (1963)** para estos sistemas.

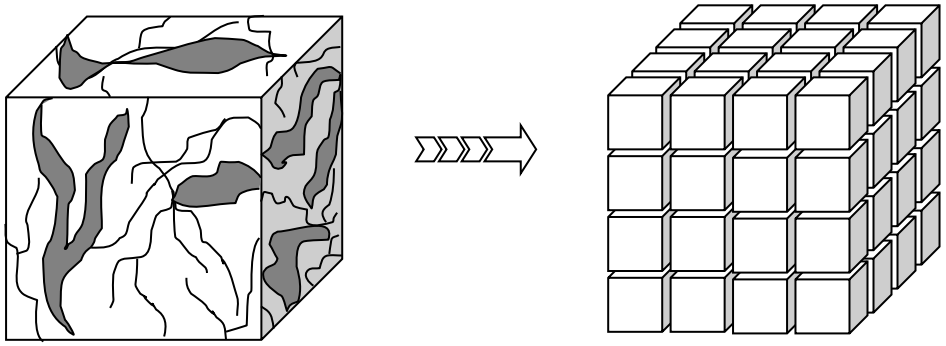


Fig. II.1 Idealización de un yacimiento naturalmente fracturado

La idea de doble porosidad la desarrollaron **Warren y Root (1963)** para flujo monofásico y la extendieron **Kazemi y cols. (1976)** a sistemas con flujo multifásico, incluyendo fuerzas capilares y gravitacionales, a partir de estos trabajos se puede desarrollar un simulador de yacimientos naturalmente fracturados. Este simulador considera el flujo de aceite, gas y agua en un yacimiento naturalmente fracturado bajo condiciones isotérmicas; no contempla intercambio de masa entre el agua y los hidrocarburos, además, se realiza en tres dimensiones, con una formulación totalmente implícita, empleando el método de diferencias finitas y el modelo se puede usar para simular la explotación del yacimiento.

El yacimiento naturalmente fracturado se conceptualiza como la combinación de un sistema discontinuo de bloques de matriz, inmerso en un sistema continuo de fracturas, el flujo primario en la formación ocurre dentro de las fracturas con un intercambio local de

fluidos entre ellas y los bloques de matriz. Suponemos que cada bloque tiene propiedades y forma geométrica conocida y todos los bloques que se consideran dentro de una celda de la malla se asumen idénticos. En la **Fig. II.2** se muestra de forma esquemática la relación entre los sistemas de fracturas y de bloques de matriz.

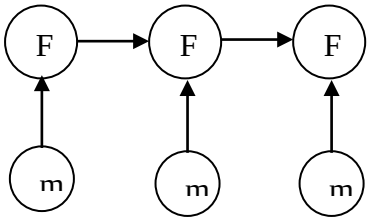


Fig. II.2 Representación de la interacción de los sistemas de fracturas (F) y matriz (m)

II.2. Formulación Matemática de las Ecuaciones de Flujo

Las ecuaciones que describen el flujo multifásico en un yacimiento naturalmente fracturado considerando el modelo de doble porosidad, comprende el conjunto siguiente de ecuaciones en cada uno de los medios (fracturas y bloques de matriz) son:

- Tres ecuaciones de flujo en el medio poroso (una para cada fase)
- Dos expresiones de presión capilar que relacionan las presiones del aceite, gas y agua
- Una ecuación de restricción para las saturaciones de las fases

Esto da un total de seis ecuaciones para cada uno de los medios, con seis incógnitas que son las presiones y las saturaciones de las fases aceite, gas y agua, p_o, p_g, p_w, s_o, s_g y s_w, del medio (fractura o matriz).

A continuación se presentan las ecuaciones diferenciales que describen el flujo de cada una de las fases en el sistema de las fracturas en forma vectorial, empleando el operador nabla; el desarrollo matemático para obtener estas ecuaciones se muestra en el apéndice A.

II.2.1. Ecuaciones para las Fracturas

Las ecuaciones para el flujo multifásico en yacimientos de aceite negro en el medio continuo (fracturas) consideran el flujo de fluidos compresibles en un medio ligeramente compresible e isotrópico de doble porosidad, y son, para cada una de las fases, las siguientes:

Aceite

$$\nabla \cdot \left[\frac{b_o k k_{ro}}{\mu_o} \left(\nabla p_o - \gamma_o \nabla D \right) \right] + \left[b_o q_o \right] + \tau_{omf} = \frac{\partial}{\partial t} \left[\phi \left(b_o S_o \right) \right], \tag{2.1}$$

Gas

$$\nabla\cdot\left[\hat{R}_s b_o \frac{kk_{ro}}{\mu_o}\left(\nabla p_o - \gamma_o \nabla D\right) + b_g \frac{kk_{rg}}{\mu_g}\left(\nabla p_g - \gamma_g \nabla D\right)\right] + \left[\hat{R}_s b_o q_o + b_g q_g\right] + \hat{R}_s \tau_{omf} + \tau_{gmf} = \frac{\partial}{\partial t}\left[\phi\left(\hat{R}_s b_o S_o + b_g S_g\right)\right], \quad (2.2)$$

Agua

$$\nabla\cdot\left[\frac{kk_{rw}}{B_w \mu_w}\left(\nabla p_w - \gamma_w \nabla D\right)\right] + b_w q_w + \tau_{wmf} = \frac{\partial}{\partial t}\left[\phi b_w S_w\right], \quad (2.3)$$

El término de producción e inyección en las ecuaciones se considera por unidad de volumen de roca; en la Ec. (2.2) se considera el gas disuelto en el aceite mientras que en la Ec. (2.3) el agua no contiene gas.

En flujo multifásico el concepto de capilaridad es importante y entre los factores que afectan este concepto están: la tensión interfacial entre los fluidos y la roca, la mojabilidad de la roca, la geometría del espacio poroso y la historia de saturaciones. En este trabajo, se desprecian los efectos de la composición de las fases en la tensión interfacial y consecuentemente sobre la presión capilar. Las ecuaciones para las presiones capilares de los sistemas gas/aceite y agua/aceite, son respectivamente:

$$p_{cgo}\left(S_g\right) = p_g - p_o, \quad (2.4)$$

$$p_{cwo}\left(S_w\right) = p_o - p_w, \quad (2.5)$$

La ecuación de restricción de saturaciones de las fases aceite, gas y agua, que complementa a las ecuaciones de las fracturas es:

$$S_o + S_g + S_w = 1, \quad (2.6)$$

En las Ecs (2.1) y (2.2), los términos τ_{omf} y τ_{gmf}, representan el ritmo de intercambio de masa entre la matriz y las fracturas de las fases aceite y gas respectivamente, a condiciones de yacimiento por unidad de volumen de roca. En la Ec. (2.3), el término τ_{wmf} representa el ritmo de intercambio de masa de la fase agua entre los sistemas a las mismas condiciones.

II.2.2. Ecuaciones para la Matriz

El conjunto de ecuaciones que modelan la transferencia de fluidos de los bloques de matriz hacia las fracturas, considerando el modelo de doble porosidad de **Warren y Root (1963)** extendido por **Kazemi y cols. (1976)**, es el siguiente:

Aceite

$$-\{\tau_{omf}\} = \frac{\partial}{\partial t}[\phi(b_o S_o)]_m, \tag{2.7}$$

Gas

$$-\left\{\hat{R}_s \tau_{omf} + \tau_{gmf}\right\} = \frac{\partial}{\partial t}\left[\phi\left(\hat{R}_s b_o S_o + b_g S_g\right)\right]_m, \tag{2.8}$$

Agua

$$-\tau_{wmf} = \frac{\partial}{\partial t}[\phi b_w S_w]_m, \tag{2.9}$$

Las ecuaciones de presión capilar para los sistemas gas/aceite y agua/aceite dentro de la matriz son:

$$P_{cgom}(S_{gm}) = p_{gm} - p_{om}, \tag{2.10}$$

$$P_{cwom}(S_{wm}) = p_{om} - p_{wm}, \tag{2.11}$$

Finalmente, para la matriz se tiene la siguiente ecuación de restricción de saturaciones:

$$S_{om} + S_{gm} + S_{wm} = 1, \tag{2.12}$$

En las ecuaciones (2.7), (2.8) y (2.9), no existen términos de flujo para los bloques de matriz, y de acuerdo con el principio de conservación de masa, el ritmo de intercambio de fluidos matriz-fractura es igual al ritmo de acumulación de fluidos en los bloques de matriz.

II.2.3. Función de Transferencia de Fluidos Matriz-Fractura.

El ritmo de intercambio de fluidos matriz-fractura que interviene en las Eecuaciones (2.7), (2.8) y (2.9), depende de las condiciones locales de presión y saturación de los fluidos en los bloques de matriz y en las fracturas. En los diferentes trabajos publicados sobre yacimientos fracturados, el aspecto más importante y complejo para modelarlos, es la solución correcta del intercambio de fluidos entre los bloques de matriz y las fracturas que los rodean, y en este trabajo, la transferencia de fluidos entre ambos sistemas se representa de acuerdo a la extensión hecha por **Kazemi y cols. (1976)**.

Dependiendo de la fase en cuestión, la función de transferencia de fluidos matriz-fractura, en forma másica, se representa mediante las expresiones siguientes:

$$\hat{t}_{omf} = \sigma\left[\frac{\rho_o k k_{ro}}{\mu_o}\right]_{mf}\left(p_{om} - p_o\right), \tag{2.13}$$

$$\hat{t}_{gmf} = \sigma\left[\frac{\rho_g k k_{rg}}{\mu_g}\right]_{mf}\left(p_{gm} - p_g\right), \tag{2.14}$$

$$\hat{t}_{wmf} = \sigma\left[\frac{k k_{rw}}{B_w \mu_w}\right]_{mf}\left(p_{wm} - p_w\right), \tag{2.15}$$

En las Ecs. (2.13), (2.14) y (2.15), σ, es un factor de forma, que depende de la geometría de los bloques de matriz y fue introducido por **Warren y Root (1963)**, para relacionar los diferenciales de presión y el gasto. **Coats (1989)**, presentó para diferencias finitas el factor de forma como:

$$\sigma = 4\left(\frac{1}{l_x^2} + \frac{1}{l_y^2} + \frac{1}{l_z^2}\right), \tag{2.16}$$

II.2.4. Condiciones Iniciales y de Frontera

Para resolver las ecuaciones diferenciales, se deben especificar las condiciones iniciales y de frontera. Las condiciones iniciales, saturaciones y presiones se calculan empleando un procedimiento de inicialización, basado en datos de presión capilar, contactos agua/aceite y gas/aceite, y gradientes de presión. Esta inicialización asegura que el sistema se encuentra en equilibrio capilar y gravitacional al tiempo cero.

Comúnmente, las presiones de las fases se especifican a una profundidad de referencia y las relaciones de presión capilar y densidad se emplean para realizar los cálculos por medio de la ecuación diferencial ordinaria siguiente, que es resultado de suponer que los potenciales de la fase no varían horizontalmente:

$$\frac{dp_p}{dD} = \rho_p(p_p)g, \;\; p = o, w, g \quad , \quad t = 0\,, \tag{2.17}$$

Para tiempos mayores a cero, se deben especificar las condiciones de frontera; para esto, se pueden agrupar las fronteras bajo dos nombres generales: externas, que constituyen a las fronteras físicas del dominio de flujo, e internas, que son los pozos.

Para el pozo, el simulador permite dos condiciones, presión constante y gasto constante; para las fronteras externas del yacimiento pueden manejarse tres condiciones que son las dos antes mencionadas y otra de cero flujo; estas condiciones se muestran a continuación:

Gasto Constante (Frontera Tipo Neumann)

En este tipo de frontera se puede decir que se especifica un gradiente de presión, cuyo flujo es constante y normal a la frontera. De acuerdo con la ecuación de Darcy, escrita en la frontera interna, se tiene:

$$\left.\frac{dp}{dr}\right|_{r=r_w} = -\frac{q\mu}{2\pi\beta_c r_w kh} \quad , \qquad t>0\,, \tag{2.18}$$

Donde μ, r_w, k y h son específicos del problema y de la fase que corresponda, por lo que asignando el gasto q (de la fase) se establece el gradiente de presión. Para la frontera externa del yacimiento esta frontera significa un flujo constante y normal a través de ella.

Presión Constante (Frontera Tipo Dirichlet)

Para fronteras internas, esta especificación se refiere a pozos productores o inyectores a presión constante en la cara de la formación. Por otro lado, en fronteras externas, esta especificación implica que la presión en la frontera no varía, ocurriendo en yacimientos que se recargan por medio de una entrada de agua fuerte de un acuífero. En este caso se especifica la presión de la fase p en la frontera, a lo largo del tiempo, esto es:

$$p_p(\mathit{frontera}, t) = C \quad , \qquad t>0\,, \tag{2.19}$$

Cero Flujo

Esta condición es un caso particular de la condición de gasto constante (o frontera tipo Neumann), donde el flujo a través de la frontera es inexistente por lo que el gradiente de presión es cero; se representa como:

$$\left.\frac{dp}{dr}\right|_{\mathit{frontera}} = 0 \qquad\qquad t>0 \tag{2.20}$$

Un yacimiento volumétrico con las fronteras externas totalmente selladas, es equivalente a un gradiente de presión nulo.

II.3. Formulación Numérica de las Ecuaciones de Flujo

En esta sección, se presentan las ecuaciones numéricas de flujo para el modelo de doble porosidad, y se muestra la aplicación del método de Newton-Raphson a la solución de los sistemas de ecuaciones algebraicas no lineales correspondientes. También se aprovecha el esquema matricial resultante para reducirlo mediante operaciones y optimizar su solución, al requerirse menos cantidad de memoria; sin embargo, a nivel de cómputo esta reducción introduce cierta inestabilidad en la solución del sistema de ecuaciones.

Las ecuaciones que describen el comportamiento de flujo de fluidos en las fracturas del yacimiento, Ecs. (2.1) a (2.6), y las que representan el comportamiento de los bloques de matriz, Ecs. (2.7) a (2.12), son no lineales, por lo que no se pueden resolver por métodos analíticos; para solucionarlas, se recurre a métodos numéricos que dan su aproximación en diferencias finitas, se genera un sistema algebraico de ecuaciones no lineales, que puede ser resuelto mediante el método iterativo de Newton-Raphson. En cada iteración, el método de Newton-Raphson genera un sistema algebraico de ecuaciones lineales que puede resolverse mediante algún algoritmo aplicable a la solución de sistemas de ecuaciones con matrices dispersas.

Los sistemas de ecuaciones no lineales para las fracturas y los bloques de matriz, se resuelven numéricamente. El carácter continuo de estas ecuaciones, en espacio y tiempo, se cambia por un carácter discreto mediante su aproximación en diferencias finitas, el cual es uno de los métodos más usados pero que introduce dispersión numérica del orden del tamaño de la malla, por lo que a escala de campo limita su precisión para el cálculo del patrón de flujo y de la recuperación de hidrocarburos.

Los términos de flujo de las ecuaciones de las fracturas y de los bloques de matriz se aproximan mediante diferencias centrales, mientras que los términos de acumulación se aproximan mediante diferencias regresivas. Este proceso de discretización da como resultado un sistema de ecuaciones algebraicas no lineales en cada etapa de tiempo. Estas ecuaciones, son las que constituyen el simulador y están escritas en términos de operadores en diferencias como se muestra a continuación.

II.3.1. Ecuaciones de las Fracturas en Diferencias Finitas

Las ecuaciones de flujo para las fracturas, aproximadas por el método de diferencias finitas, son las siguientes:

Aceite

$$\Delta\left[T_o(\Delta p_o+\gamma_o\Delta D)\right]_{ijk}^{n+1}+\left[b_o q_o\right]_{ijk}^{n+1}+\left[\tau_{omf}\right]_{ijk}^{n+1}=\frac{V_{r,ijk}}{\Delta t}\Delta_t\left[\phi b_o(1-s_g-s_w)\right]_{ijk}, \tag{2.21}$$

Gas

$$\begin{aligned}&\Delta\left[T_g(\Delta p_g+\gamma_g\Delta D)\right]_{ijk}^{n+1}+\Delta\left[\hat{R}_s T_o(\Delta p_o+\gamma_o\Delta D)\right]_{ijk}^{n+1}+\left[b_g q_g+\hat{R}_s b_o q_o\right]_{ijk}^{n+1}+\\&\left[\tau_{gmf}+\hat{R}_s\tau_{omf}\right]_{ijk}^{n+1}=\frac{V_{r,ijk}}{\Delta t}\Delta_t\left[\phi b_g s_g+\phi b_o\hat{R}_s(1-s_g-s_w)\right]_{ijk}\end{aligned}, \tag{2.22}$$

Agua

$$\Delta\left[T_w(\Delta p_w+\gamma_w\Delta D)\right]_{ijk}^{n+1}+\left[b_w q_w\right]_{ijk}^{n+1}+\left[\tau_{wmf}\right]_{ijk}^{n+1}=\frac{V_{r,ijk}}{\Delta t}\Delta_t\left[\phi b_w s_w\right)]_{ijk}, \tag{2.23}$$

Las ecuaciones de presión capilar para los sistemas gas/aceite y agua/aceite, se discretizan de la manera siguiente:

$$P_{cgo,ijk}^{n+1}(S_g) = p_{g,ijk}^{n+1} - p_{o,ijk}^{n+1}, \tag{2.24}$$

$$P_{cwo,ijk}^{n+1}(S_w) = p_{o,ijk}^{n+1} - p_{w,ijk}^{n+1}, \tag{2.25}$$

La ecuación de restricción de saturaciones expresada en términos de diferencias finitas es:

$$\left(S_o + S_g + S_w\right)_{ijk}^{n+1} = 1, \tag{2.26}$$

De las Ecs. (2.21), (2.22) y (2.23), los términos T_o, T_g y T_w representan la transmisibilidad de las fases aceite, gas y agua, respectivamente, en las fracturas y se definen como:

$$T_o = \alpha\left(\frac{b_o k_{ro}}{\mu_o}\right), \tag{2.27}$$

$$T_g = \alpha\left(\frac{b_g k_{rg}}{\mu_g}\right), \tag{2.28}$$

$$T_w = \alpha\left(\frac{k_{rw}}{B_w \mu_w}\right), \tag{2.29}$$

Donde α, es un factor geométrico que depende del sistema de coordenadas geométricas elegidas.

Para evaluar las transmisibilidades de los fluidos en las fracturas, en las fronteras de las celdas, se utiliza el concepto corriente arriba, es decir, las propiedades dentro de los paréntesis se evalúan a las condiciones de presión y de saturación de la celda de mayor potencial.

Las aproximaciones de la función de transferencia de fluidos matriz-fractura que funciona como fuente o sumidero en la fractura, son las siguientes:

$$\left(\tau_{omf}\right)_{ijk}^{n+1} = \left[T_{omf}\left(p_{om} - p_o\right)\right]_{ijk}^{n+1}, \tag{2.30}$$

$$\left(\tau_{gmf}\right)_{ijk}^{n+1} = \left[T_{gmf}\left(p_{gm} - p_g\right)\right]_{ijk}^{n+1}, \tag{2.31}$$

$$\left(\tau_{wmf}\right)_{ijk}^{n+1} = \left[T_{wmf}\left(p_{wm} - p_w\right)\right]_{ijk}^{n+1}, \tag{2.32}$$

De estas ecuaciones observamos los términos T_{omf}, T_{gmf} y T_{wmf}, que representan la transmisibilidad de las fases entre los bloques de matriz y las fracturas, se definen como:

$$T_{omf} = \sigma V_r k_m \left(\frac{b_o k_{ro}}{\mu_o}\right)_{mf}, \tag{2.33}$$

$$T_{gmf} = \sigma V_r k_m \left(\frac{b_g k_{rg}}{\mu_g}\right)_{mf}, \tag{2.34}$$

$$T_{wmf} = \sigma V_r k_m \left(\frac{k_{rw}}{B_w \mu_w}\right)_{mf}, \tag{2.35}$$

En donde el coeficiente σ es un factor geométrico (conocido como factor de forma) que representa el área superficial de los bloques de matriz por unidad de volumen y longitud característica asociada al flujo matriz-fractura.

Para evaluar las transmisibilidades de los fluidos entre los bloques de matriz y las fracturas, se utiliza el concepto corriente arriba, es decir, las propiedades dentro del paréntesis se evalúan a las condiciones del medio de mayor potencial, que pueden ser, de las fracturas o de los bloques de matriz.

Las Ecs. (2.24), (2.25) y (2.26) pueden ser acopladas en las Ecs. (2.21), (2.22) y (2.23) para reducir el número de ecuaciones y de incógnitas del sistema, de seis a tres por celda de cálculo.

Las presiones de las fases gas y agua, p_g y p_w respectivamente, pueden ser eliminadas usando las relaciones de presión capilar, dadas en las Ecs. (2.24) y (2.25):

$$p_g = p_o + p_{cgo}, \tag{2.36}$$

$$p_w = p_o - p_{cwo}, \tag{2.37}$$

La saturación de aceite, S_o, puede ser eliminada usando la relación expresada en (2.26):

$$S_o = 1 - S_g - S_w, \tag{2.38}$$

La porosidad de cada medio en el nivel de tiempo *n+1*, se puede escribir en función de la compresibilidad de la roca, c_r, considerada como constante :

$$\phi^{n+1} = \phi^n \left[1 + c_r \left(p_o^{n+1} - p_o^n \right)\right], \tag{2.39}$$

Acoplando las expresiones de las Ecs. (2.36), (2.37), (2.38) y (2.39) en (2.21), (2.22) y (2.23), el sistema de ecuaciones reducido es el siguiente:

Aceite

$$\begin{aligned} &\Delta\left[T_o(\Delta p_o + \gamma_o \Delta D)\right]_{ijk}^{n+1} + \left[b_o q_o\right]_{ijk}^{n+1} + \left[T_{omf}(p_{om} - p_o)\right]_{ijk}^{n+1} = \\ &\frac{V_{p,ijk}^n}{\Delta t}\left\{\left[1 + c_r(p_o^{n+1} - p_o^n)\right]\left[b_o(1 - s_g - s_w)\right]^{n+1} - \left[b_o(1 - s_g - s_w)\right]^n\right\}_{ijk} \end{aligned}, \tag{2.40}$$

Gas

$$\begin{aligned} &\Delta\left[T_g(\Delta p_o + \Delta p_{cgo} + \gamma_g \Delta D)\right]_{ijk}^{n+1} + \Delta\left[\hat{R}_s T_o(\Delta p_o + \gamma_o \Delta D)\right]_{ijk}^{n+1} + \left[b_g q_g + \hat{R}_s b_o q_o\right]_{ijk}^{n+1} + \\ &\left[T_{gmf}(p_{om} - p_o + p_{cgom} - p_{cgo}) + \hat{R}_s T_{omf}(p_{om} - p_o)\right]_{ijk}^{n+1} = \\ &\frac{V_{p,ijk}}{\Delta t}\left\{\left[1 + c_r(p_o^{n+1} - p_o^n)\right]\left[b_g s_g + b_o \hat{R}_s(1 - s_g - s_w)\right]^{n+1} - \left[b_g s_g + b_o \hat{R}_s(1 - s_g - s_w)\right]^n\right\}_{ijk} \end{aligned}, \tag{2.41}$$

Agua

$$\begin{aligned} &\Delta\left[T_w(\Delta p_o - \Delta p_{cwo} + \gamma_w \Delta D)\right]_{ijk}^{n+1} + \left[b_w q_w\right]_{ijk}^{n+1} + \left[T_{wmf}(p_{om} - p_o - p_{cwom} + p_{cwo})\right]_{ijk}^{n+1} = \\ &\frac{V_{p,ijk}^n}{\Delta t}\left\{\left[1 + c_r(p_o^{n+1} - p_o^n)\right]\left[b_w s_w\right]^{n+1} - \left[b_w s_w\right]^n\right\}_{ijk} \end{aligned}, \tag{2.42}$$

El sistema definido por las Ecs. (2.40), (2.41) y (2.42) para las fracturas, constituye en cada celda *ijk* de la malla un subsistema de tres ecuaciones algebraicas no lineales, con la presión de la fase aceite p_o y las saturaciones de las fases gas y agua, S_g y S_w, como incógnitas.

II.3.2. Ecuaciones de los Bloques de Matriz en Diferencias Finitas

Siguiendo un procedimiento similar que para las fracturas, el sistema de ecuaciones diferenciales que modelan el flujo multifásico en los bloques de matriz, mostradas en las Ecs. (2.7), (2.8), y (2.9) aproximadas mediante diferencias finitas son las siguientes:

Aceite

$$-\left[\tau_{omf}\right]_{ijk}^{n+1} = \frac{V_{pm,ijk}^n}{\Delta t}\left\{\left[1 + c_r(p_o^{n+1} - p_o^n)\right]\left[b_o(1 - s_g - s_w)\right]^{n+1} - \left[b_o(1 - s_g - s_w)\right]^n\right\}_{m,ijk}, \tag{2.43}$$

Gas

$$-\left[\tau_{gmf}+\hat{R}_s\tau_{omf}\right]_{ijk}^{n+1}=$$

$$\frac{V_{pm,ijk}}{\Delta t}\left\{\left[1+c_r(p_o^{n+1}-p_o^n)\right]\left[b_g s_g+b_o\hat{R}_s(1-s_g-s_w)\right]^{n+1}-\left[b_g s_g+b_o\hat{R}_s(1-s_g-s_w)\right]^n\right\}_{m,ijk}, \quad (2.44)$$

Agua

$$-\left[\tau_{wmf}\right]_{ijk}^{n+1}=\frac{V_{pm,ijk}^n}{\Delta t}\left\{\left[1+c_r(p_o^{n+1}-p_o^n)\right]\left[b_w s_w\right]^{n+1}-\left[b_w s_w\right]^n\right\}_{m,ijk}, \quad (2.45)$$

Donde los términos de transferencia de fluidos son:

Aceite

$$-\left[\tau_{omf}\right]_{ijk}^{n+1}=-\left[T_{omf}(p_{om}-p_o)\right]_{ijk}^{n+1}, \quad (2.46)$$

Gas

$$-\left[\tau_{gmf}+\hat{R}_s\tau_{omf}\right]_{ijk}^{n+1}=-\left[T_{gmf}(p_{om}-p_o+p_{cgom}-p_{cgo})+\left(\hat{R}_s T_o\right)_{mf}(p_{om}-p_o)\right]_{ijk}^{n+1}, \quad (2.47)$$

Agua

$$-\left[\tau_{wmf}\right]_{ijk}^{n+1}=-\left[T_{wmf}(p_{om}-p_o-p_{cwom}+p_{cwo})\right]_{ijk}^{n+1}, \quad (2.48)$$

El sistema definido por las Ecs.(2.43), (2.44) y (2.45) para los bloques de matriz, constituye un subsistema de tres ecuaciones algebraicas no lineales con la presión de las fase aceite de la matriz p_{om}, las saturaciones de las fases gas y agua, S_g y S_w, como incógnitas.

Finalmente, el sistema algebraico de ecuaciones en diferencias finitas, expresado por las Ecs. (2.40), (2.41) y (2.42) para el sistema de fracturas y las Ecs. (2.43), (2.44) y (2.45) para los bloques de matriz, consiste en cada nivel de tiempo *n+1* en cada celda *ijk* de la malla de cálculo, de un conjunto de seis ecuaciones con el mismo número de incógnitas.

II.4. Solución del Sistema de Ecuaciones

El conjunto de ecuaciones en diferencias, que describe el comportamiento de flujo de fluidos en el yacimiento, Ecs. (2.40), (2.41), (2.42), (2.43), (2.44) y (2.45) constituyen un sistema algebraico de ecuaciones no lineales. Debido a esto, su solución se obtiene mediante el método iterativo de Newton-Raphson, generando un sistema lineal de ecuaciones en cada iteración.

A continuación se presenta la linealización del sistema de ecuaciones, las reducciones matriciales y el sistema resultante se resuelve empleando un método iterativo de solución de sistemas de ecuaciones lineales llamado GMRES sobre el que se dan algunos detalles.

II.4.1. Linealización del Sistema de Ecuaciones

El conjunto de ecuaciones no lineales se resuelve mediante el método Totalmente-Implícito. Como las ecuaciones de la matriz no tienen términos de flujo ya que únicamente existe el término de acumulación y el término de transferencia de fluidos, las ecuaciones son tratadas de una manera totalmente implícita de una forma más sencilla. Para esto, requerimos linealizar el sistema de ecuaciones por lo que empleamos el método de Newton Raphson, **Ertekin, Abou-Kassem y King (2001)**.

Para aplicar el método de Newton Raphson, se definen las funciones de residuos las expresiones en diferencias finitas (2.40), (2.41) y (2.42) para el sistema de fracturas y, (2.43), (2.44) y (2.45) para los bloques de matriz, dando el siguiente conjunto de ecuaciones para la fractura:

Aceite

$$\begin{aligned} &R_{o,ijk}^{n+1} = \Delta\left[T_o(\Delta p_o + \gamma_o \Delta D)\right]_{ijk}^{n+1} + \left[b_o q_o\right]_{ijk}^{n+1} + \left[T_{omf}(p_{om} - p_o)\right]_{ijk}^{n+1} - \\ &\frac{V_{p,ijk}^{n}}{\Delta t}\left\{\left[1 + c_r(p_o^{n+1} - p_o^n)\right]\left[b_o(1 - s_g - s_w)\right]^{n+1} - \left[b_o(1 - s_g - s_w)\right]^n\right\}_{ijk} = 0 \end{aligned} \quad , \tag{2.49}$$

Gas

$$\begin{aligned} &R_{g,ijk}^{n+1} = \Delta\left[T_g(\Delta p_o + \Delta p_{cgo} + \gamma_g \Delta D)\right]_{ijk}^{n+1} + \Delta\left[\hat{R}_s T_o(\Delta p_o + \gamma_o \Delta D)\right]_{ijk}^{n+1} + \\ &\left[b_g q_g + \hat{R}_s b_o q_o\right]_{ijk}^{n+1} + \left[T_{gmf}(p_{om} - p_o + p_{cgom} - p_{cgo}) + \hat{R}_s T_{omf}(p_{om} - p_o)\right]_{ijk}^{n+1} - \frac{V_{p,ijk}}{\Delta t} \quad , \\ &\left\{\left[1 + c_r(p_o^{n+1} - p_o^n)\right]\left[b_g s_g + b_o \hat{R}_s(1 - s_g - s_w)\right]^{n+1} - \left[b_g s_g + b_o \hat{R}_s(1 - s_g - s_w)\right]^n\right\}_{ijk} = 0 \end{aligned} \tag{2.50}$$

Agua

$$\begin{aligned} &R_{w,ijk}^{n+1} = \Delta\left[T_w(\Delta p_o - \Delta p_{cwo} + \gamma_w \Delta D)\right]_{ijk}^{n+1} + \left[b_w q_w\right]_{ijk}^{n+1} + \\ &\left[T_{wmf}(p_{om} - p_o - p_{cwom} + p_{cwo})\right]_{ijk}^{n+1} - \\ &\frac{V_{p,ijk}^{n}}{\Delta t}\left\{\left[1 + c_r(p_o^{n+1} - p_o^n)\right]\left[b_w s_w\right]^{n+1} - \left[b_w s_w\right]^n\right\}_{ijk} = 0 \end{aligned} \quad , \tag{2.51}$$

Mientras que para los bloques de matriz, las funciones de residuos son:

Aceite

$$R_{om,ijk}^{n+1} = -\left[T_{omf}\left(p_{om} - p_o\right)\right]_{ijk}^{n+1} - \frac{V_{pm,ijk}^{n}}{\Delta t}\left\{\left[1 + c_r\left(p_o^{n+1} - p_o^{n}\right)\right]\left[b_o\left(1 - s_g - s_w\right)\right]^{n+1} - \left[b_o\left(1 - s_g - s_w\right)\right]^{n}\right\}_{m,ijk} = 0 \quad , \tag{2.52}$$

Gas

$$R_{gm,ijk}^{n+1} = -\left[T_{gmf}\left(p_{om} - p_o + p_{cgom} - p_{cgo}\right) + \left(\hat{R}_s T_o\right)_{mf}\left(p_{om} - p_o\right)\right]_{ijk}^{n+1} - \frac{V_{pm,ijk}}{\Delta t}\left\{\left[1 + c_r\left(p_o^{n+1} - p_o^{n}\right)\right]\left[b_g s_g + b_o \hat{R}_s\left(1 - s_g - s_w\right)\right]^{n+1} - \left[b_g s_g + b_o \hat{R}_s\left(1 - s_g - s_w\right)\right]^{n}\right\}_{m,ijk} = 0 \quad , \tag{2.53}$$

Agua

$$R_{wm,ijk}^{n+1} = -\left[T_{wmf}\left(p_{om} - p_o - p_{cwom} + p_{cwo}\right)\right]_{ijk}^{n+1} - \frac{V_{pm,ijk}^{n}}{\Delta t}\left\{\left[1 + c_r\left(p_o^{n+1} - p_o^{n}\right)\right]\left[b_w s_w\right]^{n+1} - \left[b_w s_w\right]^{n}\right\}_{m,ijk} = 0 \quad , \tag{2.54}$$

Si se definen los siguientes vectores de incógnitas, o variables primarias, en la celda *ijk*, **Arana y Rodríguez (1996)**, tenemos para las fracturas:

$$\mathbf{U}_{f,ijk} = \left(p_o, S_g, S_w\right)_{ijk}^{T} , \tag{2.55}$$

Y para los bloques de matriz,

$$\mathbf{U}_{m,ijk} = \left(p_o, S_g, S_w\right)_{m,ijk}^{T} , \tag{2.56}$$

Tenemos que la dependencia de las funciones de residuos de las fracturas es:

Aceite

$$R_{o,ijk} = R_{o,ijk}\left(\mathbf{U}_{f,ijk-1}, \mathbf{U}_{f,ij-1k}, \mathbf{U}_{f,i-1jk}, \mathbf{U}_{f,ijk}, \mathbf{U}_{m,ijk}, \mathbf{U}_{f,i+1jk}, \mathbf{U}_{f,ij+1k}, \mathbf{U}_{f,ijk+1}\right) = 0 , \tag{2.57}$$

Gas

$$R_{g,ijk} = R_{g,ijk}\left(\mathbf{U}_{f,ijk-1}, \mathbf{U}_{f,ij-1k}, \mathbf{U}_{f,i-1jk}, \mathbf{U}_{f,ijk}, \mathbf{U}_{m,ijk}, \mathbf{U}_{f,i+1jk}, \mathbf{U}_{f,ij+1k}, \mathbf{U}_{f,ijk+1}\right) = 0 , \tag{2.58}$$

Agua

$$R_{w,ijk} = R_{w,ijk}\left(\mathbf{U}_{f,ijk-1}, \mathbf{U}_{f,ij-1k}, \mathbf{U}_{f,i-1jk}, \mathbf{U}_{f,ijk}, \mathbf{U}_{m,ijk}, \mathbf{U}_{f,i+1jk}, \mathbf{U}_{f,ij+1k}, \mathbf{U}_{f,ijk+1}\right) = 0 , \tag{2.59}$$

Mientras que las funciones de residuos de la matriz son dependientes de las siguientes variables:

Aceite

$$R_{om,ijk} = R_{om,ijk}\left(\mathbf{U}_{f,ijk}, \mathbf{U}_{m,ijk}\right) = 0, \tag{2.60}$$

Gas

$$R_{gm,ijk} = R_{gm,ijk}\left(\mathbf{U}_{f,ijk}, \mathbf{U}_{m,ijk}\right) = 0, \tag{2.61}$$

Agua

$$R_{wm,ijk} = R_{wm,ijk}\left(\mathbf{U}_{f,ijk}, \mathbf{U}_{m,ijk}\right) = 0, \tag{2.62}$$

El algoritmo iterativo de solución de Newton-Raphson se establece expandiendo las funciones de residuos en la iteración (ν+1) mediante el truncamiento de la serie de Taylor, alrededor del nivel iterativo conocido (ν), **Ertekin, Abou-Kassem y King (2001)**; lo anterior conduce al siguiente sistema de ecuaciones lineales para cada uno de los sistemas que conforman al yacimiento naturalmente fracturado.

Para las fracturas tenemos las siguientes ecuaciones:

$$\begin{aligned}&\sum_{j,l}\left\{\frac{\partial R^{\nu}_{p,ijk}}{\partial p_{o,j,l}}\delta p^{\nu+1}_{o,j,l} + \frac{\partial R^{\nu}_{p,ijk}}{\partial S_{g,j,l}}\delta S^{\nu+1}_{g,j,l} + \frac{\partial R^{\nu}_{p,ijk}}{\partial S_{w,j,l}}\delta S^{\nu+1}_{w,j,l}\right\} + \frac{\partial R^{\nu}_{p,ijk}}{\partial p_{om,ijk}}\delta p^{\nu+1}_{om,ijk} + \frac{\partial R^{\nu}_{p,ijk}}{\partial S_{gm,ijk}}\delta S^{\nu+1}_{gm,ijk} \\ &+ \frac{\partial R^{\nu}_{p,ijk}}{\partial S_{wm,ijk}}\delta S^{\nu+1}_{wm,ijk} = -R^{\nu}_{p,ijk}\end{aligned}, \tag{2.63}$$

Donde *p* indica la fase de hidrocarburos, que puede ser aceite o gas. Para el agua tenemos:

$$\begin{aligned}&\sum_{j,l}\left\{\frac{\partial R^{\nu}_{w,ijk}}{\partial p_{o,l}}\delta p^{\nu+1}_{o,j,l} + \frac{\partial R^{\nu}_{w,ijk}}{\partial S_{w,l}}\delta S^{\nu+1}_{w,j,l}\right\} + \frac{\partial R^{\nu}_{w,ijk}}{\partial p_{om,ijk}}\delta p^{\nu+1}_{om,ijk} \\ &+ \frac{\partial R^{\nu}_{w,ijk}}{\partial S_{wm,ijk}}\delta S^{\nu+1}_{wm,ijk} = -R^{\nu}_{w,ijk}\end{aligned}, \tag{2.64}$$

De las Ecs. (2.63) y (2.64) tenemos al subíndice *l* que representa al conjunto de celdas de la malla *ijk-1*, *ij-1k*, *i-1jk*, *ijk*, *i+1jk*, *ij+1k* y *ijk+1*.

Para los bloques de matriz tenemos:

$$\frac{\partial R^{\nu}_{pm,ijk}}{\partial p_{o,ijk}}\delta p^{\nu+1}_{o,ijk}+\frac{\partial R^{\nu}_{pm,ijk}}{\partial S_{g,ijk}}\delta S^{\nu+1}_{g,ijk}+\frac{\partial R^{\nu}_{pm,ijk}}{\partial S_{w,ijk}}\delta S^{\nu+1}_{w,ijk}+\frac{\partial R^{\nu}_{pm,ijk}}{\partial p_{o,m,ijk}}\delta p^{\nu+1}_{om,ijk}+\frac{\partial R^{\nu}_{pm,ijk}}{\partial S_{gm,ijk}}\delta S^{\nu+1}_{gm,ijk}+\frac{\partial R^{\nu}_{pm,ijk}}{\partial S_{wm,ijk}}\delta S^{\nu+1}_{wm,ijk}=-R^{\nu}_{pm,ijk}, \tag{2.65}$$

Donde *p* indica la fase de hidrocarburos, que puede ser aceite o gas. Para el agua tenemos:

$$\frac{\partial R^{\nu}_{wm,ijk}}{\partial p_{o,ijk}}\delta p^{\nu+1}_{o,ijk}+\frac{\partial R^{\nu}_{wm,ijk}}{\partial S_{w,ijk}}\delta S^{\nu+1}_{w,ijk}+\frac{\partial R^{\nu}_{wm,ijk}}{\partial p_{om,ijk}}\delta p^{\nu+1}_{om,ijk}+\frac{\partial R^{\nu}_{wm,ijk}}{\partial S_{wm,ijk}}\delta S^{\nu+1}_{wm,ijk}=-R^{\nu}_{wm,ijk}, \tag{2.66}$$

De las Ecs. (2.63) a (2.66), aplicadas a las tres, se tiene un sistema de ecuaciones, donde las incógnitas son los cambios iterativos de las presiones y saturaciones en cada una de las celdas, tanto en la fractura como en la matriz: $(\delta p_o \delta S_g \delta S_w)^{\nu+1}_{ijk}$, y $(\delta p_o \delta S_g \delta S_w)^{\nu+1}_{m,ijk}$, respectivamente, donde $i=1,2,...,I$, $j=1,2,...,J$, $k=1,2,\ldots,K$. En particular, $\delta p^{\nu+1}_{o,ijk} = p^{\nu+1}_{o,ijk} - p^{\nu}_{o,ijk}$; las incógnitas restantes se definen de forma similar.

Con el fin de simplificar la escritura, el superíndice correspondiente al nivel de tiempo *n+1* fue eliminado de las Ecs. (2.63) a (2.66).

El proceso iterativo del nivel de tiempo *n+1* se inicia comúnmente con la siguiente estimación de la solución:

Para las fracturas:

$$\left(p_o, S_g, S_w\right)^{0}_{ijk} = \left(p_o, S_g, S_w\right)^{n}_{ijk}, \tag{2.67}$$

Para los bloques de matriz:

$$\left(p_o, S_g, S_w\right)^{0}_{m,ijk} = \left(p_o, S_g, S_w\right)^{n}_{m,ijk}, \tag{2.68}$$

Termina cuando los cambios iterativos de las incógnitas de la fractura y la matriz son menores a una tolerancia predefinida, ε, de esta manera tenemos por ejemplo, $\left|\delta p^{\nu+1}_{o,ijk}\right| < \varepsilon$; las incógnitas restantes se definen de igual manera, tanto para las fracturas como para los bloques de matriz.

El sistema lineal de ecuaciones obtenido mediante la aplicación del método iterativo de Newton-Raphson en la iteración (ν+1), Ecs. (2.63) a (2.66), puede escribirse en forma compacta como sigue:

$$\left[\boldsymbol{J}\right]^{\nu}\delta\boldsymbol{U}^{\nu+1}=-R^{\nu}, \tag{2.69}$$

Donde [$\boldsymbol{J}$] se conoce como la matriz Jacobiana, $\boldsymbol{\delta U}$ es el vector de incógnitas y R es el vector de residuos.

La estructura matricial del subsistema de ecuaciones para la celda *ijk*, es la siguiente:

$$\begin{aligned}
&\begin{bmatrix} E_{ff} & [0] \\ [0] & [0] \end{bmatrix}^{\nu}_{i-1,jk}\begin{bmatrix}\delta\boldsymbol{U}_f \\ \delta\boldsymbol{U}_m\end{bmatrix}^{\nu+1}_{i-1,jk}+\begin{bmatrix} G_{ff} & [0] \\ [0] & [0] \end{bmatrix}^{\nu}_{i,j-1,k}\begin{bmatrix}\delta\boldsymbol{U}_f \\ \delta\boldsymbol{U}_m\end{bmatrix}^{\nu+1}_{i,j-1,k}+\\
&\begin{bmatrix} C_{ff} & [0] \\ [0] & [0] \end{bmatrix}^{\nu}_{ij,k-1}\begin{bmatrix}\delta\boldsymbol{U}_f \\ \delta\boldsymbol{U}_m\end{bmatrix}^{\nu+1}_{ij,k-1}+\begin{bmatrix} A_{ff} & A_{fm} \\ A_{mf} & A_{mm} \end{bmatrix}^{\nu}_{ijk}\begin{bmatrix}\delta\boldsymbol{U}_f \\ \delta\boldsymbol{U}_m\end{bmatrix}^{\nu+1}_{i,j,k}+\\
&\begin{bmatrix} B_{ff} & [0] \\ [0] & [0] \end{bmatrix}^{\nu}_{ij,k+1}\begin{bmatrix}\delta\boldsymbol{U}_f \\ \delta\boldsymbol{U}_m\end{bmatrix}^{\nu+1}_{ij,k+1}+\begin{bmatrix} F_{ff} & [0] \\ [0] & [0] \end{bmatrix}^{\nu}_{i,j+1,k}\begin{bmatrix}\delta\boldsymbol{U}_f \\ \delta\boldsymbol{U}_m\end{bmatrix}^{\nu+1}_{i,j+1,k}+\\
&\begin{bmatrix} D_{ff} & [0] \\ [0] & [0] \end{bmatrix}^{\nu}_{i+1,jk}\begin{bmatrix}\delta\boldsymbol{U}_f \\ \delta\boldsymbol{U}_m\end{bmatrix}^{\nu+1}_{i+1,jk}=-\begin{bmatrix}R_f \\ R_m\end{bmatrix}^{\nu}_{ijk}
\end{aligned}, \tag{2.70}$$

Donde E_{ff}, G_{ff}, C_{ff}, A_{ff}, A_{fm}, A_{mf}, A_{mm}, B_{ff}, F_{ff} y D_{ff} son submatrices de orden (3)× (3). Las submatrices A_{ff} y A_{fm}, contienen las derivadas de las funciones de residuos de las fracturas con respecto a las incógnitas de la fractura y de la matriz en la celda *ijk*, respectivamente. Las submatrices A_{mf} y A_{mm}, contienen las derivadas de las funciones de residuos de los bloques de matriz con respecto a las incógnitas de la fractura y de la matriz en la celda *ijk*, respectivamente. Las submatrices E_{ff}, G_{ff}, C_{ff}, B_{ff}, F_{ff} y D_{ff} contienen las derivadas de las funciones de residuos de las fracturas con respecto a las incógnitas de las fracturas en las celdas *i-1jk*, *ij-1k*, *ijk-1*, *ijk+1*, *ij+1k* e *i+1jk*, respectivamente. Además, δU_f y δU_m, son los vectores de incógnitas de la fractura y la matriz, de la celda en cuestión, finalmente R_f y R_m representan a las funciones de residuos de las fases para cada uno de los sistemas.

II.4.2 Reducción Matricial del Sistema de Ecuaciones

La estructura matricial del sistema lineal de ecuaciones (2.70), permite reducirlo a un sistema de tres ecuaciones con tres incógnitas por celda, similar al que se obtiene en el caso de yacimientos homogéneos de aceite negro. La reducción del sistema de ecuaciones se realiza mediante el complemento de Schur, aplicado por **Arana y Rodríguez (1996)**.

Por medio de esta reducción, se logra disminuir el tamaño de la matriz completa con lo que reducimos los requerimientos de memoria de cómputo y el número de operaciones al tener una menor cantidad de elementos.

La estructura matricial del sistema de ecuaciones mostrado en la Ec. (2.70) permite llevar a cabo una reducción del problema matricial acoplando las ecuaciones de la matriz en las ecuaciones de la fractura; para esto, el subsistema de ecuaciones, (2.70), puede escribirse de la manera siguiente:

Tenemos para las ecuaciones de las fracturas:

$$\begin{aligned} &E^{\nu}_{ff,i-1\,jk}\delta U^{\nu+1}_{f,i-1\,jk} + G^{\nu}_{ff,ij-1k}\delta U^{\nu+1}_{f,ij-1k} + C^{\nu}_{ff,ijk-1}\delta U^{\nu+1}_{f,ijk-1} + A^{\nu}_{ff,ijk}\delta U^{\nu+1}_{f,ijk} + \\ &A^{\nu}_{fm,ijk}\delta U^{\nu+1}_{m,ijk} + B^{\nu}_{ff,ijk+1}\delta U^{\nu+1}_{f,ijk+1} + F^{\nu}_{ff,ij+1k}\delta U^{\nu+1}_{f,ij+1k} + D^{\nu}_{ff,i+1\,jk}\delta U^{\nu+1}_{f,i+1\,jk} = -R^{\nu}_{f,ijk} \end{aligned}, \tag{2.71}$$

Para las ecuaciones de los bloques de matriz:

$$A^{\nu}_{mf,ijk}\delta \boldsymbol{U}^{\nu+1}_{f,ijk} + A^{\nu}_{mm,ijk}\delta \boldsymbol{U}^{\nu+1}_{m,ijk} = -\boldsymbol{R}^{\nu}_{m,ijk}, \tag{2.72}$$

Para la reducción, de la Ec. (2.72), se resuelve para las incógnitas de la matriz $\delta U^{\nu+1}_{m,ijk}$, escribiéndola en términos de las incógnitas de la fractura $\delta \boldsymbol{U}^{\nu+1}_{f,ik}$, lo que nos da como resultado:

$$\delta \boldsymbol{U}^{\nu+1}_{m,ijk} = -\boldsymbol{R}^{*\nu}_{m,ijk} - A^{*\nu}_{mf,ijk}\delta \boldsymbol{U}^{\nu+1}_{f,ijk}, \tag{2.73}$$

Donde,

$$A^{*\nu}_{mf,ijk} = \left[A^{\nu}_{mm,ijk}\right]^{-1} A^{\nu}_{mf,ijk}, \tag{2.74}$$

$$R^{*\nu}_{m,ijk} = \left[A^{\nu}_{mm,ijk}\right]^{-1} R^{\nu}_{m,ijk}, \tag{2.75}$$

La Ec. (2.73), representa la solución de las incógnitas en la matriz, en función de las incógnitas de la fractura.

Sustituyendo la Ec. (2.73), en la ecuación de las fracturas, se tiene:

$$\begin{aligned} &E^{\nu}_{ff,i-1\,jk}\delta U^{\nu+1}_{f,i-1\,jk} + G^{\nu}_{ff,ij-1k}\delta U^{\nu+1}_{f,ij-1k} + C^{\nu}_{ff,ijk-1}\delta U^{\nu+1}_{f,ijk-1} + A^{\nu}_{ff,ijk}\delta U^{\nu+1}_{f,ijk} + \\ &A^{\nu}_{fm,ijk}\left[-R^{*\nu}_{m,ijk} - A^{*\nu}_{mf,ijk}\delta U^{\nu+1}_{f,ijk}\right] + B^{\nu}_{ff,ijk+1}\delta U^{\nu+1}_{f,ijk+1} + F^{\nu}_{ff,ij+1k}\delta U^{\nu+1}_{f,ij+1k} + \\ &D^{\nu}_{ff,i+1\,jk}\delta U^{\nu+1}_{f,i+1\,jk} = -R^{\nu}_{f,ijk} \end{aligned}, \tag{2.76}$$

Agrupando términos:

$$\begin{aligned} &E^{\nu}_{ff,i-1\,jk}\delta U^{\nu+1}_{f,i-1\,jk} + G^{\nu}_{ff,ij-1k}\delta U^{\nu+1}_{f,ij-1k} + C^{\nu}_{ff,ijk-1}\delta U^{\nu+1}_{f,ijk-1} + A^{*\nu}_{ff,ijk}\delta U^{\nu+1}_{f,ijk} + \\ &B^{\nu}_{ff,ijk+1}\delta U^{\nu+1}_{f,ijk+1} + F^{\nu}_{ff,ij+1k}\delta U^{\nu+1}_{f,ij+1k} + D^{\nu}_{ff,i+1\,jk}\delta U^{\nu+1}_{f,i+1\,jk} = -R^{*\nu}_{f,ijk} \end{aligned}, \tag{2.77}$$

En donde tenemos:

$$A^{*\nu}_{ff,ijk} = A^{\nu}_{ff,ijk} - A^{\nu}_{fm,ijk} A^{*\nu}_{mf,ijk}, \tag{2.78}$$

$$R^{*\nu}_{f,ijk} = -R^{\nu}_{f,ijk} + A^{\nu}_{fm,ijk} R^{*\nu}_{m,ijk}, \tag{2.79}$$

El subsistema de ecuaciones (2.77), consiste de tres ecuaciones con tres incógnitas por cada celda de la malla de cálculo cuyas incógnitas son únicamente las del sistema de fracturas $\delta \boldsymbol{U}^{\nu+1}_{f,ijk}$.

La estructura matricial del subsistema de ecuaciones reducido, posee una estructura matricial similar a la que se obtiene para un yacimiento homogéneo. Por medio del acoplamiento de las ecuaciones, los requerimientos de cómputo son ligeramente mayores a los correspondientes al caso de yacimientos no fracturados.

II.5. Tratamiento de Pozos

En la simulación numérica de yacimientos, un punto importante es el pronóstico de los gastos de los pozos y/o la presión de fondo fluyendo, de forma aproximada. El manejo de pozos en los simuladores, presenta dificultades que requieren consideraciones especiales mencionadas por **Ertekin, Abou-Kassem y King (2001)**, éstas son:

1. El bloque que contiene la terminación del pozo generalmente es más grande comparado con el tamaño del pozo, por lo que la presión del bloque calculada por el simulador es una pobre estimación de la presión de fondo fluyendo.
2. El acoplamiento de la compleja interacción entre el yacimiento y el pozo es en ocasiones problemática, particularmente, en el caso de pozos multilaterales.
3. La asignación de los gastos de producción de cada fase en flujo multifásico, es complicada cuando el gasto de una de ellas o la producción total del pozo es especificada.

Otros problemas se encuentran cuando varios pozos se encuentran en una celda de la malla, y cuando un pozo no se encuentra en el centro del bloque, además, el tratamiento de un pozo se vuelve más complicado cuando consideramos su comportamiento de flujo, detalles de terminación, estimulación e hidráulica del sistema.

Como se mencionó anteriormente, los pozos se consideran fronteras internas del yacimiento, que pueden ser a presión especificada (frontera tipo Dirichlet) o de gasto especificado (frontera tipo Neumann).

Con base en lo anterior, los simuladores de yacimientos requieren de una relación funcional entre la presión del bloque y la presión de fondo del pozo para calcular la presión de fondo fluyendo cuando se asigna el gasto, o para calcular el gasto cuando se asigna la presión de fondo fluyendo; a estas relaciones las conocemos como modelo de pozo y deben considerar las características geométricas del pozo (radio, localización en el bloque,

inclinación, etc.) y las propiedades del yacimiento en la vecindad del pozo (daño, anisotropía, saturación, etc.).

II.5.1. Modelo de Pozo

El modelado de pozos juega un rol importante en la simulación de yacimientos, ya que la precisión en el cálculo del gasto o de la presión del pozo se relaciona directamente a este modelo. Comúnmente los modelos usados se basan en la solución analítica de presión en 2D para flujo radial y en un adecuado cálculo del índice de productividad. La principal dificultad del modelado es el problema de singularidad por la diferencia de escala entre el pequeño diámetro del pozo y el bloque de malla de grandes dimensiones usado en la simulación, y también, otra dificultad es la naturaleza radial del flujo alrededor del pozo.

Existen varios modelos de pozo publicados, por lo que es conveniente mencionar algunos de ellos, sin embargo, el que se usa en este trabajo por su amplio uso y conocimiento de resultados, con el fin de comparación, es de Peaceman.

Diversos autores han presentado modelos de pozo y aplicaciones en simuladores numéricos como **Mrosovsky y Ridings (1974)**, quienes presentaron en su trabajo el acoplamiento de un modelo radial en dos dimensiones a un simulador tridimensional en coordenadas cartesianas. **Babu y cols. (1991)**, publicaron una ecuación analítica para calcular el radio efectivo en el bloque del pozo, el cual es necesitado para relacionar la presión del bloque con la del pozo, la ecuación es general y válida para pozos horizontales y verticales en para cualquier ubicación del pozo. **Ding y Renard (1994)**, desarrollaron una nueva representación de pozos, aplicable a mallas no uniformes y pozos no aislados, basados en el método de volumen finito, posteriormente, **Ding, Renard y Weill (1998)**, mostraron una nueva representación de pozos en simulación de yacimientos y enseñaron como implementarlo fácilmente en simuladores existentes por medio del índice de productividad.

Peaceman (1978), publicó su primer artículo relacionado al modelo de pozo basado en que la presión calculada para el bloque, es la misma que la presión de fondo fluyendo a un radio equivalente, r_o, empleando un patrón de cinco puntos y bloques cuadrados de malla; demostró que para un medio isotrópico cuadrado, que contiene un pozo productor e inyector, la presión del bloque, p_o, se relaciona con la presión de fondo fluyendo, p_{wf}, como:

$$q = \frac{2\pi\beta_c k_H h(p_o - p_{wf})}{\mu B \ln(r_o / r_w)}, \tag{2.80}$$

Esta ecuación es aplicable para pozos verticales donde p_o es considerada la presión de fondo fluyendo estacionaria a un radio $r_o = 0.2\Delta x$ en una malla cuadrada. Esta relación ha sido aceptada universalmente y ha remplazado las ecuaciones empleadas antes de que se publicara.

Posteriormente, **Peaceman (1983)**, presentó un segundo trabajo con ecuaciones para calcular valores de r_o cuando el bloque es de forma rectangular y/o la formación es anisotrópica. Adicionalmente, estudió el problema de flujo de pozos en estado estacionario con pozos productores e inyectores colocados en las esquinas. Empleando soluciones numéricas para una sola fase y con un patrón de cinco puntos, mostró que el radio equivalente (donde la presión en estado estacionario del yacimiento es igual a la presión de bloque del pozo) para pozos en bloques rectangulares con permeabilidad anisotrópica, se obtiene con:

$$r_0 = 0.28 \frac{\left[\left(\frac{k_y}{k_x} \right)^{\frac{1}{2}} \Delta x^2 + \left(\frac{k_x}{k_y} \right)^{\frac{1}{2}} \Delta y^2 \right]^{\frac{1}{2}}}{\left(\frac{k_y}{k_x} \right)^{\frac{1}{4}} + \left(\frac{k_x}{k_y} \right)^{\frac{1}{4}}}, \tag{2.81}$$

Mientras que para un caso especial de permeabilidad isotrópica en el plano horizontal ($k_x = k_y$), el radio equivalente es:

$$r_0 = 0.14\left(\Delta x^2 + \Delta y^2\right), \tag{2.82}$$

La relación de presión de bloque y de fondo fluyendo, Ec. (2.80), puede ser introducida al simulador empleando el Índice de Productividad del pozo, *WI*, como se muestra en la Ec. (2.83) para una sola fase.

$$q_{well,ijk} = WI\left(p_{b,ijk} - p_{wf,ijk}\right), \tag{2.83}$$

Donde *WI*, se define como:

$$WI = \frac{2\pi\beta_c k_H h}{\mu B \ln(r_o / r_w)}, \tag{2.84}$$

Y $k_H = \sqrt{k_x k_y}$. Escrito de otra forma, tenemos al índice de productividad

$$WI = \frac{1}{\mu B} G_w, \tag{2.85}$$

Donde, $G_w = \frac{2\pi\beta_c k_H h}{\ln(r_o / r_w)}$, es un factor geométrico del pozo.

Para cada bloque disparado la magnitud de WI se requiere para relacionar el gasto del pozo con la presión del bloque.

Para flujo multifásico se tiene:

$$q_{well,ijk} = \frac{k_{r,p}}{\mu_p B_p} G_w \left(p_{b,ijk} - p_{wf,ijk} \right), \tag{2.86}$$

Donde:

p	=	fase aceite, gas o agua (o, g, w)
$k_{r,p}$	=	permeabilidad relativa a la fase p
μ_p	=	viscosidad de la fase p
B_p	=	factor de volumen de la fase p
$p_{b,ijk}$	=	presión del bloque i,j,k
$P_{wf,ijk}$	=	presión de fondo fluyendo de la celda i,j,k
$q_{well,ijk}$	=	gasto del pozo de la fase p
G_w	=	Factor geométrico del pozo

Las propiedades de los fluidos se evalúan a las condiciones de mayor potencial; adicionalmente, de la Ec. (2.86) se obtiene la expresión siguiente para la presión de fondo fluyendo:

$$p_{wf,ijk} = p_{b,ijk} - \frac{q_{well,ijk,p}}{WI} = p_{b,ijk} - \frac{q_{well,ijk,p} \mu_p B_p}{G_w k_{r,p}}, \tag{2.87}$$

II.5.2. Solución del Modelo de Pozo

Para mantener la estabilidad del simulador se requiere manejar el modelo del pozo de manera implícita ya que otras metodologías, en algunos casos, acarrean o producen oscilaciones físicamente irreales a causa de la forma de evaluar a las incógnitas. Éstas, se presentan cuando pequeños cambios en los parámetros de flujo, ocasionan grandes variaciones en los gastos de producción/inyección. Para mejorar la estabilidad de la formulación totalmente implícita, se necesita desarrollar un método para evaluar las incógnitas al tiempo $n+1$, y al nivel iterativo $\nu+1$, para esto se expande la ecuación del pozo por medio de la serie de Taylor, lo que nos da:

$$q_{sc_k}^{\nu+1} = q_{sc_k}^{\nu} + \left.\frac{\partial q_{sc_k}}{\partial p_{o_k}}\right|^{\nu} \delta p_{o_k}^{\nu+1} + \left.\frac{\partial q_{sc_k}}{\partial s_{g_k}}\right|^{\nu} \delta s_{g_k}^{\nu+1} + \left.\frac{\partial q_{sc_k}}{\partial s_{w_k}}\right|^{\nu} \delta s_{w_k}^{\nu+1} + \left.\frac{\partial q_{sc_k}}{\partial p_{wf_{ref}}}\right|^{\nu} \delta p_{wf_{ref}}^{\nu+1}, \tag{2.88}$$

Donde todos los términos se encuentran al nivel de tiempo $n+1$ y el subíndice k representa a la celda disparada del intervalo. La ecuación (2.88) introduce siete incógnitas $q_{osc_k}^{n+1}, q_{wsc_k}^{n+1}, q_{gsc_k}^{n+1}, p_{o_k}^{n+1}, s_{g_k}^{n+1}, s_{w_k}^{n+1}$ y $p_{wf_{ref}}^{n+1}$ al nivel iterativo $\nu+1$.

Para resolver estas incógnitas, se tienen solo seis ecuaciones, la ecuación (2.88) escrita para el aceite, gas y agua (tres ecuaciones), y las ecuaciones de balance de materia de las tres fases, por lo que se busca una manera de eliminar una de las incógnitas a través de las especificaciones del pozo.

II.5.2.1. Presión Especificada de Pozo

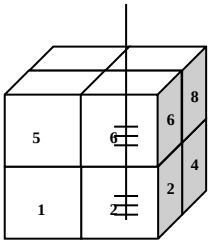
Para pozos con presión especificada, la presión en la cara del pozo, p_{wfref}, al final de la iteración $\upsilon+1$ debe ser igual a la presión al principio de la iteración υ. Esto es:

$$p_{wf_{ref}}^{\upsilon+1} = p_{wf_{ref}}^{\upsilon} = p_{wfsp}\,, \tag{2.89}$$

Sustituyendo la Ee. (2.89) en (2.88):

$$q_{sc_k}^{\upsilon+1} = q_{sc_k}^{\upsilon} + \left.\frac{\partial q_{sc_k}}{\partial p_{o_k}}\right|^{\upsilon} \delta p_{o_k}^{\upsilon+1} + \left.\frac{\partial q_{sc_k}}{\partial s_{g_k}}\right|^{\upsilon} \delta s_{g_k}^{\upsilon+1} + \left.\frac{\partial q_{sc_k}}{\partial s_{w_k}}\right|^{\upsilon} \delta s_{w_k}^{\upsilon+1}, \tag{2.90}$$

Donde ya no aparece la incógnita de la presión de fondo fluyendo. De esta manera, se ha eliminado ésta incógnita del conjunto de ecuaciones. El objetivo del método implícito, es el mejorar la estabilidad de la formulación en diferencias finitas, para ello, se incluye la ecuación (2.90) en el sistema de ecuaciones formado por las ecuaciones de flujo. En la **Fig. II.3** se muestra de forma esquemática el sistema resultante (jacobiano), donde **X** representa submatrices de tercer orden con elementos diferentes de cero para una malla de con dos celdas en la dirección de "*x*", dos en la dirección "*y*" y dos en la dirección de "*z*", con un pozo perforando las celdas dos y seis.



P1	P2	P3	P4	P5	P6	P7	P8	Q1	Q2
X	X	X	0	X	0	0	0	0	0
X	X	0	X	0	X	0	0	X	0
X	0	X	X	0	0	X	0	0	0
0	X	X	X	0	0	0	X	0	0
X	0	0	0	X	X	X	0	0	0
0	X	0	0	X	X	0	X	0	X
0	0	X	0	X	0	X	X	0	0
0	0	0	X	0	X	X	X	0	0
0	X	0	0	0	0	0	0	X	0
0	0	0	0	0	X	0	0	0	X

Fig. II.3 Sistema matricial considerando un pozo con presión especificada

Se debe notar cuando se especifica la presión de fondo fluyendo, los gastos de las fases de cada una de las celdas se convierten en incógnitas del sistema (Q1 y Q2).

II.5.2.2. Gasto Especificado de Pozo

Para pozos con gasto especificado, lo establecemos para todo el pozo (para una fase en particular o total), para esto, escribimos la Ec. (2.88) para cada bloque perforado y se suman las ecuaciones obtenidas para tener el gasto total del pozo, esto es,

$$\sum_k q_{sc_k}^{\upsilon+1} = \sum_k q_{sc_k}^{\upsilon} + \sum_k \left[\left. \frac{\partial q_{sc_k}}{\partial p_{o_k}} \right|^{\upsilon} \delta p_{o_k}^{\upsilon+1} + \left. \frac{\partial q_{sc_k}}{\partial s_{g_k}} \right|^{\upsilon} \delta s_{g_k}^{\upsilon+1} + \left. \frac{\partial q_{sc_k}}{\partial s_{w_k}} \right|^{\upsilon} \delta s_{w_k}^{\upsilon+1} + \left. \frac{\partial q_{sc_k}}{\partial p_{wf_{ref}}} \right|^{\upsilon} \delta p_{wf_{ref}}^{\upsilon+1} \right], \qquad (2.91)$$

Para un pozo con gasto especificado, el ritmo de producción/inyección al final de la iteración ν+1 debe ser igual al gasto al principio de la iteración ν como se muestra en la Ec. (2.92).

$$\sum_k q_{sc_k}^{\upsilon+1} = \sum_k q_{sc_k}^{\upsilon} = q_{spsc}\,, \qquad (2.92)$$

Sustituyendo la Ec. (2.92) en (2.91), obtenemos:

$$\sum_k q_{sc_k}^{\upsilon} + \sum_k \left[\left. \frac{\partial q_{sc_k}}{\partial p_{o_k}} \right|^{\upsilon} \delta p_{o_k}^{\upsilon+1} + \left. \frac{\partial q_{sc_k}}{\partial s_{g_k}} \right|^{\upsilon} \delta s_{g_k}^{\upsilon+1} + \left. \frac{\partial q_{sc_k}}{\partial s_{w_k}} \right|^{\upsilon} \delta s_{w_k}^{\upsilon+1} + \left. \frac{\partial q_{sc_k}}{\partial p_{wf_{ref}}} \right|^{\upsilon} \delta p_{wf_{ref}}^{\upsilon+1} \right] = 0\,, \qquad (2.93)$$

Rearreglando (2.93), obtenemos una expresión para la presión de fondo fluyendo de referencia en la Ec. (2.94):

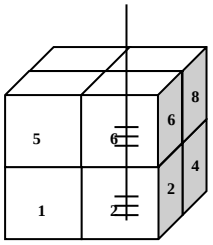
$$p_{wf_{ref}}^{\upsilon+1} = p_{wf_{ref}}^{\upsilon} - \frac{\sum_k \left[\left. \frac{\partial q_{sc_k}}{\partial p_{o_k}} \right|^{\upsilon} \delta p_{o_k}^{\upsilon+1} + \left. \frac{\partial q_{sc_k}}{\partial s_{g_k}} \right|^{\upsilon} \delta s_{g_k}^{\upsilon+1} + \left. \frac{\partial q_{sc_k}}{\partial s_{w_k}} \right|^{\upsilon} \delta s_{w_k}^{\upsilon+1} + \left. \frac{\partial q_{sc_k}}{\partial p_{wf_{ref}}} \right|^{\upsilon} \delta p_{wf_{ref}}^{\upsilon+1} \right]}{\sum_k \left. \frac{\partial q_{sc_k}}{\partial p_{wf_{ref}}} \right|^{\upsilon}}\,, \qquad (2.94)$$

La **Fig. II.4** muestra la matriz de coeficientes generada por el método totalmente implícito para un pozo de gasto especificado donde cada uno de los elementos diferentes de cero representa una submatriz de tercer orden.

Cuando especificamos el gasto para el manejo de pozos, la incógnita que se calcula es la presión de fondo fluyendo de referencia de cada uno de los pozos y a partir de ella se puede establecer la presión por cada celda a partir de la Ec. (2.95).

$$p_{wf_k} = p_{wfref} + \int_{H_{ref}}^{H_k} \gamma_{wb} dH \tag{2.95}$$

Donde p_{wfk} y p_{wfref} son las presiones de fondo fluyendo en la celda k y de referencia, γ_{wb} es el gradiente de presión, el cual es función de la densidad del fluido y, H_{ref} y H_k son las profundidades de la celda k y de referencia; esta ecuación se resuelve de forma iterativa con el método de Newton-Raphson para p_{wfk} y γ_{wb}.



P1	P2	P3	P4	P5	P6	P7	P8	Pwf
X	X	X	0	X	0	0	0	0
X	X	0	X	0	X	0	0	X
X	0	X	X	0	0	X	0	0
0	X	X	X	0	0	0	X	0
X	0	0	0	X	X	X	0	0
0	X	0	0	X	X	0	X	X
0	0	X	0	X	0	X	X	0
0	0	0	X	0	X	X	X	0
0	X	0	0	0	X	0	0	X

Fig. II.4 Sistema matricial considerando un pozo con gasto especificado

III. PROGRAMACIÓN EN PARALELO

Las implementaciones numéricas de diversos modelos de ingeniería requieren, para resolverse, de tiempo y recursos computacionales de gran magnitud, pero se ven limitados por el poder computacional disponible; por esta razón, se ha optado por investigar en otras tecnologías de cómputo, donde la programación en paralelo ha tomado un lugar preponderante. La idea básica del paralelismo es que un conjunto de computadoras trabajan en cooperación para resolver una tarea o problema, esto es, la tarea se divide entre los procesadores disponibles. La programación en paralelo, además de ofrecer soluciones rápidas, puede resolver grandes problemas y más complejos, donde los datos de entrada o resultados intermedios exceden la capacidad de memoria de una computadora, por otro lado, las simulaciones pueden realizarse con mejor resolución y los fenómenos físicos pueden modelarse de una manera más realista. Los algoritmos empleados en la programación en paralelo, son más complejos y robustos en comparación a los empleados en programación serial por lo que si los comparamos trabajando sobre un solo procesador el algoritmo paralelo sería más lento. Esta técnica de programación implica una curva de aprendizaje y su aplicación en un tipo de problema, de acuerdo a su naturaleza, puede ocasionar resultados buenos o malos.

Pancake (1996), establece que la naturaleza del problema es la clave del éxito o el fracaso de la programación en paralelo, en particular, en los patrones de acceso a los datos y el cómputo asociado. De los problemas que pueden resolverse empleando el cómputo paralelo; algunos autores establecen que los cálculos en conjuntos de datos son totalmente independientes, por lo que el procesamiento se puede realizar en máquinas independientes; otro tipo de problemas intercambian parte de la información obtenida al finalizar los procesos en cada una de las máquinas y para un funcionamiento correcto necesitan un buen balance de carga de tareas; hay un tipo de problemas que requiere una sincronización baja de los procesos, en la que cuando un proceso termina su trabajo debe esperar a que los demás concluyan para poder compartir resultados y continuar con su trabajo; y finalmente, algunos problemas requieren sincronización total de los procesos que se ejecutan en paralelo, de tal forma que para avanzar durante la solución del problema se requiere que las tareas o procesos terminen completamente e intercambien información, presentando un mayor esfuerzo de programación.

En este capítulo se incluye una introducción a la programación en paralelo, mostrando algunos conceptos útiles para el entendimiento claro de las ideas planteadas en el trabajo; se presenta el tipo de arquitecturas de computadoras paralelas existentes y sus características, se muestran los paradigmas, modelos de programación y las bases para el diseño de aplicaciones para estos sistemas y se enseña la descripción y justificación de las herramientas empleadas durante el desarrollo.

III.1. Conceptos de Cómputo Paralelo

La programación en paralelo es un área con mucho potencial de desarrollo y para lograr entrar en ella se requiere el conocimiento de algunos conceptos para su fácil comprensión, así como de características y propiedades que se deben de tomar en cuenta para diseñar las aplicaciones. A continuación se presentan algunos de estos elementos:

III.1.1. Conceptos Fundamentales

Programa Concurrente

Hansen (1978) y **Foster (1995)**, mencionan que estos programas consisten de un número fijo de procesos secuenciales, que se ejecutan simultáneamente. Un proceso define sus propias variables, algunos procedimientos comunes y una instrucción de inicio.

Descomposición del Problema

Un aspecto principal de la programación en paralelo es la descomposición del problema; esto se refiere a dividirlo en una serie de problemas del mismo tipo, pero más pequeños que pueden resolverse de manera independiente y a partir de las soluciones particulares, obtener la solución del problema original.

Modularidad

La modularidad de un programa significa que varios componentes se desarrollaron de forma separada (módulos) y se combinan para completar un programa. Las interacciones entre módulos se restringen con interfaces bien definidas; por lo tanto, las implementaciones pueden cambiar sin necesidad de modificar otros componentes.

Determinismo

Un algoritmo o programa es determinista si la ejecución de una entrada en particular de datos, alcanza siempre la misma salida; de no ser así el algoritmo es no determinista.

Paso de Mensajes

Este concepto se refiere a la acción de comunicar información entre procesadores; consiste de dos pasos: movimiento de los datos al buffer y envío de los datos. En general, existen tres formas de paso de mensajes para un sistema de memoria distribuida: síncrona, asíncrona y de operaciones globales. El paso de mensajes síncrono requiere que todos los mensajes se completen antes de que los cálculos puedan continuar. El paso asíncrono de mensajes, por otro lado, permite un poco de traslape de procesamiento y comunicaciones. Operaciones globales de paso de mensajes distribuyen los datos de un procesador a otro.

Localidad

Indica una alta razón de acceso a memoria local con respecto a accesos de memoria remota. Es la llave para un alto desempeño en arquitecturas de multicomputadoras.

Contención de Memoria

Es el conflicto de memoria que ocurre cuando varios procesadores acceden la memoria central, compartida en modo de vector.

Latencia

Para los datos que se transfieren de un procesador a otro, se requiere establecer la ruta entre ellos. La asignación de esta ruta y la identificación de las localizaciones implicadas en la transferencia introducen un retraso, el cual se conoce como latencia. Para reducirla se disminuye el número de comunicaciones entre procesadores empacando varios mensajes en uno solo, si se hace esto, el factor limitante de la comunicación se asocia con el ancho de banda o la velocidad a la que los datos serán transferidos, una vez que el link entre procesadores se ha establecido.

Ancho de Banda

Este concepto se refiere a la cantidad de datos que se pueden transmitir en un periodo de tiempo; es decir, la tasa de transferencia máxima permitida por el sistema.

Granularidad

En cómputo paralelo, es la medida cualitativa de la relación de cómputo y comunicación; dicho de otra manera, si un problema se subdivide en varias tareas paralelas, cada una de ellas debe ser lo más larga posible o maximizada, para reducir la sobrecarga por comunicación entre procesadores a una cifra insignificante. Esta puede ser de las formas siguientes:

Coarse (gruesa): donde grandes cantidades de trabajo de cómputo se realizan entre las comunicaciones.

Fine (fina): donde pequeñas cantidades de trabajo de cómputo se realizan entre las comunicaciones.

III.1.2. Elementos de Diseño

Balance de Carga

Dentro de un sistema distribuido algunos de los nodos pueden estar altamente cargados de tareas, mientras que otros tienen poco trabajo computacional por realizar. Esta diferencia de nivel de carga sugiere que es posible mejorar el comportamiento del sistema reubicando algunas tareas, de manera que se maximice el uso de los recursos; esta mejora del sistema se conoce como balance de carga la cual consiste en la asignación de tareas a los procesadores del sistema, de tal manera que cada uno de ellos realice el mayor trabajo posible, minimizando el tiempo de ejecución; sin embargo, si se tienen muchas tareas pequeñas, provocarán una menor granularidad y un incremento en la sobrecarga.

El balance de carga puede realizarse de manera estática o dinámica. En el balance estático de carga, las tareas (y datos en sistemas de memoria distribuida) se asignan a procesadores al principio de la ejecución, esto significa que la transferencia de carga se hace una sola vez, ya que se conoce el trabajo a realizar y puede dividirse. En el balance de carga dinámico, las tareas (y datos) se asignan a procesadores conforme se realiza el

cómputo, lo que implica un ajuste continuo del nivel de carga de tareas en cada nodo. Para este tipo de balance de carga se emplea el concepto de "conjunto de tareas", donde cada procesador obtiene la tarea siguiente cuando está listo para ejecutarla. El balance de carga dinámico puede implementarse más eficientemente en sistemas de memoria compartida que en sistemas de memoria distribuida, ya que la transferencia de datos entre memorias locales puede ser parte de la asignación de tareas; desafortunadamente, la sobrecarga asociada al balance de carga dinámico debida al envío de mensajes puede ser importante, por lo que se vuelve un balance entre la maximización del uso de los recursos y la minimización del envío de mensajes; además, para obtener un buen comportamiento en paralelo se requiere equilibrio entre la granularidad de tareas y el balance de carga.

El balance de carga dinámico como lo menciona **Anguille y cols. (1995)**, generalmente se divide en tres tipos de estrategias: emisor iniciado (EI), receptor iniciado (RI) e intercambio periódico (IP) . La primera de las estrategias (EI) consiste en que los nodos altamente cargados de tareas busquen nodos con baja carga para transferir o enviar algunas de sus tareas. Por el contrario en la segunda estrategia (RI) los nodos con baja carga buscan aquellos con altas cargas, de los que adquieren o reciben tareas. En las estrategias de IP, los procesadores intercambian su estado del sistema y usan la información de carga del último periodo como referencia, para decidir la forma de transferir el trabajo en el periodo actual.

Las estrategias EI y RI se consideran generalmente como las técnicas más poderosas en el balance de carga dinámico y han sido comparadas demostrando que la estrategia EI debe usarse en sistemas con carga de ligera a moderada y el método RI debe emplearse en sistemas con cargas altas. Esta comparación usualmente se relaciona a la cantidad de sobrecarga requerida para enviar mensajes entre los diferentes procesadores.

Comunicación y Sincronización

En un sistema de memoria distribuida, la comunicación se debe al intercambio de datos necesario entre los procesadores durante el cómputo, lo que genera sobrecarga. En sistemas de memoria compartida existen retrasos, que dan el mismo efecto como el tiempo de comunicación con procesadores que están detenidos mientras esperan los datos necesarios para poder continuar.

La sincronización es necesaria cuando ciertas partes del cómputo deben realizarse antes de que todo el proceso pueda continuar; se refiere a la coordinación de tareas paralelas en tiempo real. Existen dos aspectos de sincronización que contribuyen a la sobrecarga, el primero es el tiempo empleado para efectuar la comunicación, ya que requiere que los procesadores realicen verificaciones. El segundo aspecto se refiere a que algunos procesadores pueden suspender sus tareas y esperar para continuar posteriormente con sus tareas. Existen varios caminos para implementar la sincronización. Una situación común se presenta cuando se tiene una sección crítica de código, la cual es una parte secuencial de todo el programa.

Aunque la sincronización, comunicación y retrasos de memoria son diferentes, su efecto en el cómputo es el mismo, presentando un retraso mientras los datos están listos para continuar con su cómputo.

Sobrecarga

El mezclar información de un procesador a otro para optimizar el rendimiento total del sistema, requiere del uso de estructuras de paso de mensajes. El tiempo empleado en este paso de mensajes se conoce como sobrecarga. Este es un tiempo extra transcurrido sumado al tiempo de cómputo. Como resultado, todas las políticas de balance de carga deben considerarse como un balance entre el uso de los recursos y la minimización de la sobrecarga.

Puede deberse a factores como:

- Tiempo de inicio de tareas.
- Sincronización.
- Comunicación de datos.
- Sobrecarga de software debida a compiladores en paralelo, librerías, herramientas, sistema operativo, etc.
- Tiempo de terminación de tareas.

La sobrecarga es difícil de manejar y minimizar y en ocasiones limita las capacidades del balance de carga. Básicamente, las estrategias de balance dinámico de carga, que tienen que reaccionar a las perturbaciones del sistema, requieren una mayor cantidad de paso de mensajes que las técnicas de balance estático de carga.

Aceleración y Eficiencia

La medición inmediata para evaluar el comportamiento de un sistema consiste en analizar cada componente del tiempo total de ejecución:

$$T = T_{computo} + T_{comunicación} + T_{\text{sin } actividad} \ , \tag{3.1}$$

Para comparar diferentes algoritmos, es conveniente normalizar el tiempo de ejecución, o emplear métricas que puedan evaluarse usando el tiempo de CPU, como la aceleración y eficiencia.

La aceleración (S_p) es una métrica que expresa el beneficio relativo de resolver el problema en paralelo; es una medida donde se compara un algoritmo paralelo ejecutado en *p* procesadores, contra el mejor algoritmo secuencial del mismo problema; esto es:

$$S_p = \frac{tiempo\ de\ ejecución\ del\ a\lg oritmo\ serial\ más\ rápido}{tiempo\ de\ ejecución\ del\ a\lg oritmo\ paralelo\ en\ p\ procesadores} \ , \tag{3.2}$$

Sin embargo, el mejor algoritmo secuencial es difícil de identificar, por lo que se remplaza por el tiempo de ejecución del algoritmo paralelo en un procesador, con lo que obtenemos una aceleración relativa:

$$S_p = \frac{tiempo\ de\ ejecución\ usando\ un\ procesador}{tiempo\ de\ ejecución\ usando\ p\ procesadores}\ , \tag{3.3}$$

La eficiencia (E_p), expresa en forma relativa que tan ocupados están los procesadores; caracteriza la efectividad con la que un algoritmo usa los recursos computacionales y se define como:

$$E_p = \frac{S_p}{p}\ , \tag{3.4}$$

Cuando $S_p \leq p$, tenemos $E_p \leq 1$ mientras que una eficiencia $E_p = 1$ corresponde a la aceleración perfecta de $S_p = p$.

Escalabilidad del Sistema Paralelo

Para una arquitectura paralela dada y un problema de tamaño fijo, la aceleración (S_p) no se incrementa de manera indefinida conforme aumenta el número de procesadores. En varios casos, la aceleración cercana a p puede alcanzarse incrementando el tamaño del problema, siendo un indicador de la escalabilidad del algoritmo con respecto a una arquitectura paralela.

La escalabilidad denota la habilidad de demostrar un incremento proporcional de velocidad en paralelo con la adición de más procesadores. Algunos factores que contribuyen a la escalabilidad son:

- Hardware (en particular la memoria), ancho de banda y red de comunicaciones.
- Algoritmo de aplicación.
- Sobrecarga.
- Características de la aplicación y código.

Ley de Amdahl

En el cómputo en paralelo puede esperarse que si un algoritmo se ejecuta en p procesadores, la aceleración debe ser cercana a P, pero todo algoritmo tiene componentes secuenciales que la limitan eventualmente. La ley de Amdahl establece que el tiempo de cómputo es la suma de dos componentes: un componente serial S en el que los cálculos se realizan de manera consecutiva, y un componente paralelo P donde los cálculos se hacen de forma simultánea. Por lo anterior, en una máquina con un procesador el tiempo total para una tarea es:

$$T_1 = S + P\ , \tag{3.5}$$

Suponiendo que no se pierde tiempo en la sincronización, comunicación, etc., al correr en *p* procesadores, el tiempo para la misma tarea será:

$$T_p = S + \frac{P}{p} \ , \tag{3.6}$$

donde el subíndice *p* indica el tiempo en el procesador *p* de la máquina paralela. A partir de lo anterior tenemos que la aceleración es:

$$\frac{T_1}{T_p} = \frac{S+P}{S+\frac{P}{p}} \ , \tag{3.7}$$

se considera que el trabajo serial más el paralelo representa el 100%, tenemos de la ley de Amdahl cuanta aceleración (S_p) puede alcanzar un programa en paralelo:

$$S_p = \frac{1}{s+\frac{p}{n}} \ , \tag{3.8}$$

donde,

s = fracción serial del programa, incluyendo sobrecarga de comunicación.
p = fracción paralela del programa
n = número de procesadores en paralelo

La aceleración se refiere a la velocidad a la que un programa en paralelo puede ejecutarse en relación a un solo nodo o al comportamiento en serial.

Ley de Gustafson

La ley de Amdahl puede dar información pesimista, pero Gustafson presentó una formulación alternativa que enfatiza la importancia del tamaño del problema; esto es, conforme el tamaño del problema aumenta se pueden obtener eficiencias en paralelo significativas incrementando el número de celdas de cómputo. Esta ley puede representarse como:

$$S(P) = P - \alpha * (P-1) \ , \tag{3.9}$$

donde *P* es el número de procesos, *S* es la velocidad y α es la parte no paralelizable del proceso. En aplicaciones de gran escala, α es en ocasiones un número pequeño.

La ley de Amdahl asume un tamaño fijo del problema y el tamaño de la sección secuencial es independiente del número de procesadores, mientras que la Ley de Gustafson no considera estas suposiciones.

III.2. Arquitecturas de Computadoras Paralelas

En general, una computadora paralela es cualquier conjunto de elementos de proceso conectados por algún tipo de red de comunicación. La manera en que la computadora (o conjunto de computadoras) está controlada y la forma de compartir información tiene un gran impacto, influyendo en el comportamiento de los resultados y también en el nivel de esfuerzo requerido en la paralelización de una aplicación.

La arquitectura de computadoras es el estudio de la organización e interconexión de los componentes del sistema de cómputo. Las computadoras pueden construirse con memorias, unidades aritméticas, elementos de procesamiento y de almacenamiento, dando lugar a máquinas de diversos tamaños y el comportamiento funcional de sus componentes es similar entre ellas; la memoria del sistema realiza funciones de almacenamiento, la unidad central de proceso realiza operaciones y la interfaz de salida o entrada transfiere datos del procesador a un lugar apropiado, **Roosta (1999)**. Las mayores diferencias entre ellas se encuentran en la forma en la que los módulos están conectados, en las características de comportamiento de los módulos y en el método a través del cual se controla el sistema.

III.2.1. Clasificación

La arquitectura de computadoras paralelas depende de la interacción que existe entre la cantidad de tareas que pueden ejecutarse simultáneamente y la forma en que las computadoras pueden acceder a una sección particular de memoria; a partir de esto, de la clasificación de Flynn existen cuatro clases de computadoras paralelas: "*Single Instruction and Multiple Data*" (SIMD una instrucción y múltiples datos), "*Multiple Instruction and Multiple Data*" (MIMD múltiples instrucciones y múltiples datos), "*Single Instruction Single Data"* (SISD una instrucción y un conjunto de datos) y "*Multiple Instruction Single Data"* (MISD múltiples instrucciones y un conjunto de datos).

De acuerdo a **Golub y Ortega (1997),** y **Cismaşiu (2002)**, la clase SIMD se refiere a sistemas en los que todos los procesadores se encuentran controlados por uno maestro y realizan la misma instrucción (o nada) en un tiempo dado; la clase MIMD concierne a sistemas formados por un conjunto de computadoras y cada una de ellas puede operar independientemente de las demás; esto significa que los procesadores corren bajo el control de su propio programa, permitiendo gran flexibilidad en las tareas para que los procesadores las realicen en cualquier momento, lo que presenta un problema de sincronización; la clase SISD corresponde a máquinas de un procesador, donde una sola cadena de instrucciones se aplica de forma serial a un solo conjunto de datos; y finalmente la clase MISD se refiere a computadoras donde cada procesador ejecuta diferentes instrucciones en los mismos datos; esta clase prácticamente no existe y la implementación más cercana a ella es una computadora de procesamiento vectorial, con una línea de instrucciones.

Las dos primeras clases mencionadas, SIMD y MIMD, son las que comúnmente se emplean en sistemas de cómputo paralelo y en las que se enfoca este capítulo. La primera es simple para diseñar, pero requiere de hardware específico y alcanza un comportamiento óptimo sólo en algunas aplicaciones y la segunda es más flexible en su construcción pero

más compleja para el diseño, en la **Fig. III.1** se observa de manera esquemática cada una de las clases mencionadas.

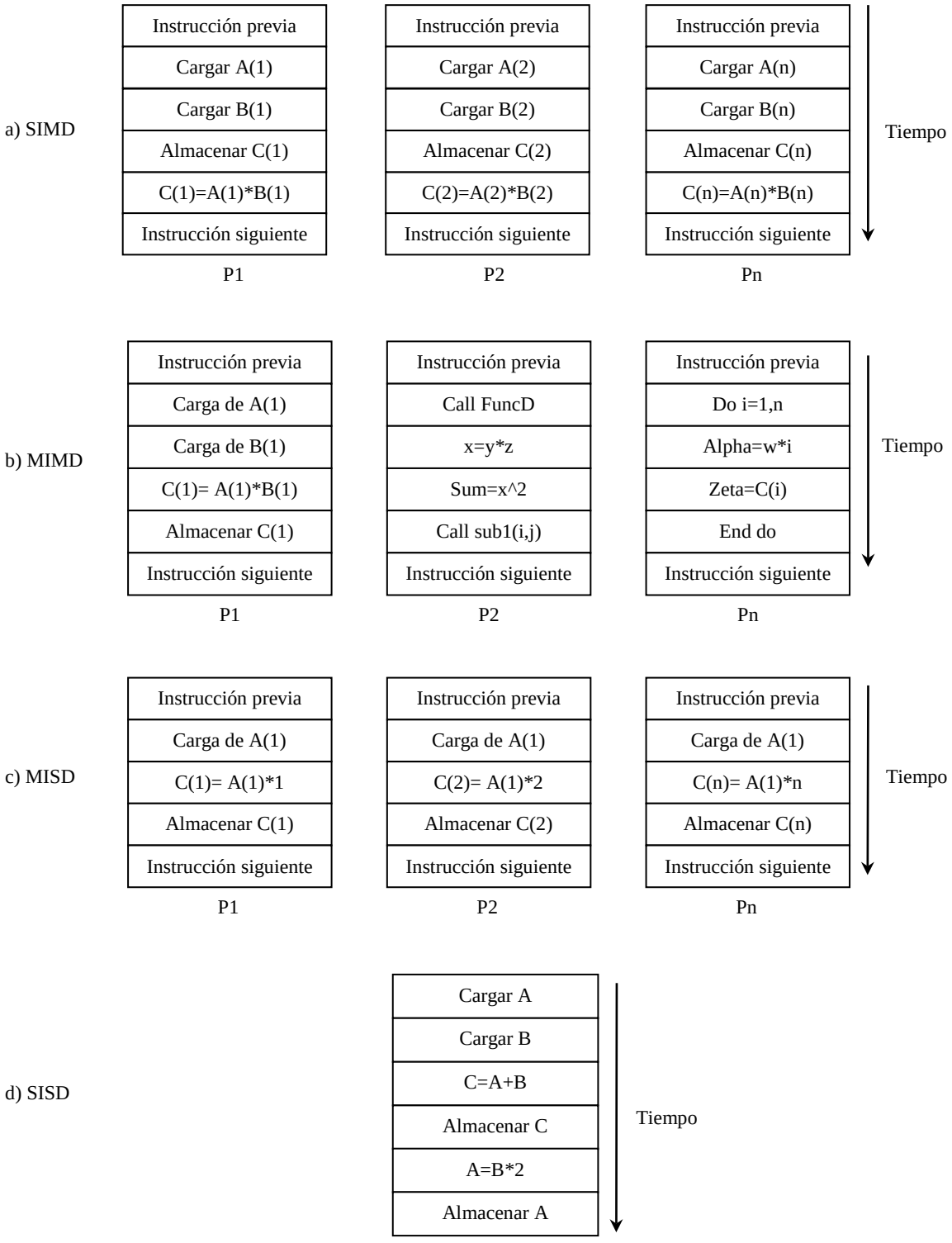


Fig. III.1 Clasificación de arquitecturas de computadoras paralelas

III.2.2. Arquitecturas Paralelas

Existe una gran cantidad de arquitecturas paralelas que si bien poseen esquemas de trabajo diferentes, pueden agruparse de acuerdo a la clasificación que se presenta en este trabajo, a continuación se muestran algunas de las más usadas.

SIMD (Single Instruction and Multiple Data)

En una multicomputadora SIMD todos los procesadores ejecutan la misma instrucción (en un tiempo dado) y poseen una sola unidad de control para ellos (procesador maestro); el procesador maestro emite instrucciones a los otros procesadores para realizar las tareas y se lleva a cabo la sincronización. En estos sistemas existe una sola instrucción operando en una secuencia de datos múltiples (una por cada procesador); como se puede observar en la figura III.1a. Estas máquinas son relativamente fáciles de programar y emplean la memoria eficientemente.

Este tipo de máquinas tienen un distribuidor de instrucciones, una red interna con un amplio ancho de banda; se prefiere para problemas especializados que se caracterizan por un alto grado de regularidad, tal como el procesamiento de imágenes.

Shared-Memory

Esta arquitectura corresponde a máquinas MIMD, donde cada procesador tiene su propia unidad de control y en cualquier momento durante la ejecución, diferentes procesadores pueden ejecutar diferentes instrucciones. En computadoras de memoria compartida (Shared-Memory) los procesadores interactúan accediendo áreas de memoria común, por esto, la aplicación en paralelo requiere de seguros para proteger información de áreas de memoria que en un instante de la ejecución no se modifiquen y también se requiere la verificación constante de estas áreas, para decidir si se puede o no acceder a ellas.

La memoria compartida debe usarse para datos y resultados que se emplean por más de un procesador. La mayor ventaja de este tipo de arquitectura se encuentra en su rapidez para la comunicación de datos entre procesadores y la principal desventaja se tiene cuando diferentes procesadores quieren usar la misma sección de memoria simultáneamente ocasionando un retraso hasta que la memoria es liberada, dicho retraso aumenta conforme el número de procesadores aumenta. Típicamente, la memoria compartida ha sido utilizada para sistemas con un número de hasta 16 procesadores. En la **Fig. III.2** se muestra de forma conceptual como está compuesto este tipo de arquitectura.

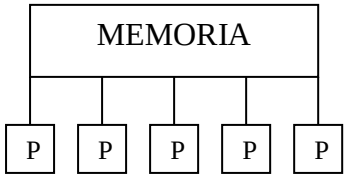


Fig. III.2 Sistema de memoria compartida (Shared Memory)

En este tipo de arquitectura, múltiples procesadores pueden operar de forma independiente pero comparten los mismos recursos de memoria y los cambios en una ubicación de memoria debido a un procesador, son visibles para todos los procesadores.

Las ventajas de estos sistemas son que el espacio global de direcciones provee una perspectiva de programación de memoria amigable al usuario y el intercambio de datos entre tareas es rápido y uniforme debido a la proximidad de la memoria y los procesadores.

Las desventajas son la falta de escalabilidad entre memoria y procesadores, ya que el aumento de procesadores puede incrementar el tráfico en la ruta de memoria compartida, también se tiene una alta responsabilidad del programador para construir estructuras de sincronización que aseguren el acceso correcto a la memoria global y el diseño con un mayor número de procesadores de máquinas con memoria compartida es difícil y costoso.

Distributed-Memory

En computadoras de memoria distribuida (Distributed-Memory) cada una de ellas ejecuta sus propias instrucciones y además poseen memoria privada, su estructura se muestra en la **Fig. III.3**; pertenece a la arquitectura del tipo MIMD. El concepto de memoria distribuida se ha vuelto el más utilizado para el diseño de computadoras paralelas ofreciendo procesamiento en paralelo al conectar decenas o cientos de computadoras por medio de una red o switch.

La clave en este tipo de arquitectura se encuentra en la habilidad de comunicarse rápidamente entre procesadores; las aplicaciones comparten información enviando mensajes entre procesadores de manera explícita, la cual se emplea en cualquier momento de la ejecución, lo que lleva a un aspecto que puede ocasionar problemas y es el tiempo requerido para transferir datos entre computadoras, el cual es un tiempo inactivo de algunos procesadores ya que para continuar con su trabajo tienen que esperar hasta recibir la información que se los permita. La desventaja de estas computadoras radica en la dificultad de manejar los recursos eficientemente.

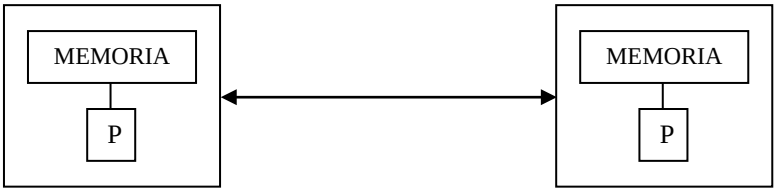


Fig. III.3 Sistema de Memoria Distribuida (Distributed Memory)

Dentro de las ventajas de estos sistemas se encuentra el que la memoria es escalable con el número de procesadores lo que significa que el incremento de procesadores y memoria son proporcionales, además, cada procesador puede acceder a su propia memoria sin generar sobrecarga y el costo es efectivo.

Las desventajas son que el programador es responsable de los detalles asociados con la comunicación de datos entre procesadores, es difícil mapear estructuras de datos (basados en memoria global) a este tipo de organización y los tiempos de acceso de memoria no son uniformes.

SMP (Symmetric Multiprocessor)

Estas computadoras se conocen como máquinas de multiprocesadores simétricos, su arquitectura es de la forma MIMD y se crean acoplando computadoras con memoria compartida. Están compuestas por múltiples procesadores homogéneos los cuales comparten acceso a una memoria en común y la palabra "*simétricos*" se refiere al hecho de que cada computadora puede recuperar datos almacenados en cualquier área de memoria en la misma cantidad de tiempo. En estos sistemas, los procesadores comparten también el bus de entrada/salida.

III.2.3. Esquemas de Comunicación

Dentro de los tipos de arquitectura mencionados se debe observar la manera en que los procesadores se comunican entre ellos, lo que se denomina como *esquemas de comunicación*. Para **Golub y Ortega (1997)**, éstos son particularmente importantes para sistemas de memoria distribuida, pero también lo son para sistemas de memoria compartida. Los esquemas existentes son:

Completamente Conectado

En un sistema completamente conectado cada procesador tiene conexión directa con todos los procesadores. Esto es, teóricamente es el esquema de conexión ideal pero es impráctico para un gran número de procesadores *p* ya que requiere *p-1* líneas de conexión para cada procesador.

Switches

Otra aproximación a completamente conectado es a través de un *switch* cruzado (como se muestra en la **Fig. III.4**) en el que cada procesador, *p*, puede ser conectado a cada memoria, *M*, a través del uso de switches. Esto tiene la ventaja de permitir a cada procesador acceder cualquier memoria con sólo un pequeño número de líneas de conexión.

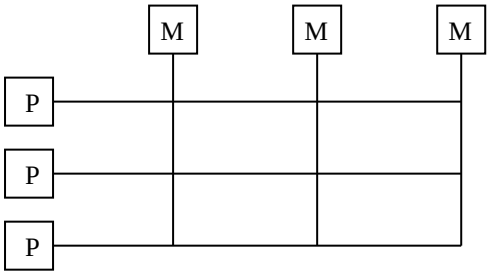


Fig. III.4 Switch de conexión cruzado

La desventaja es el que se requiere p^2 switches para conectar p procesadores con p memorias. Esto es impráctico para un número grande de procesadores pero puede ser mitigado empleando una red de switches.

Conexión de Malla

Una de las interconexiones más populares ha sido el tener a cada procesador conectado a sólo unos cuantos procesadores vecinos. En el caso de un arreglo lineal, cada uno de los procesadores están conectados con otros dos procesadores, excepto los procesadores de los extremos que se conectan con solo un procesador.

La mayoría de los arreglos de conexión de malla han utilizado un patrón de conexión de dos dimensiones, que se muestra en la **Fig. III.5**, en donde cada uno de los procesadores se encuentra conectado con sus vecinos de forma horizontal y vertical.

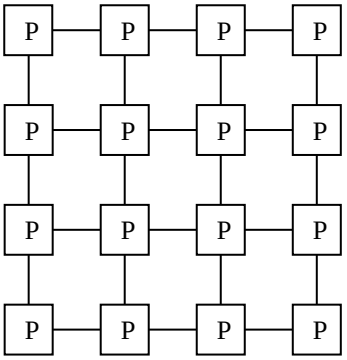


Fig. III.5 Arreglo de conexión de malla

La desventaja de este patrón de conexión se encuentra en que para transmitir datos entre procesadores distantes, estos deben pasar a través de una serie de procesadores intermedios.

Conexión Hipercubo

Una variación del principio de conexión de malla es usar, conceptualmente, conexiones locales en grandes dimensiones. Consideramos primero un cubo en tres dimensiones e imaginamos que los procesadores se localizan en los vértices del cubo. Los ejes del cubo son las vías de comunicación entre procesadores, tal que cada uno de ellos se conecta con sus tres vecinos más cercanos. Imaginando ahora este esquema empleando un cubo en k dimensiones, cada procesador se conecta con k vértices adyacentes a lo largo de los ejes del cubo, esto se conoce como conexión hipercubo y se muestra en la **Fig. III.6** para un caso particular.

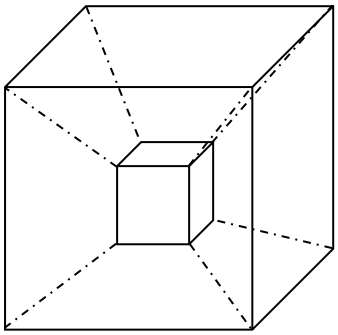


Fig. III.6. Conexión Hipercubo

En este esquema de conexión, el número de conexiones de cada procesador crece conforme el número de procesadores se incrementa.

Clusters

El esquema de cluster se muestra en la **Fig. III.7**, y aquí hay *n* clusters que consisten de *m* procesadores. Dentro de cada cluster, los procesadores están conectados de alguna manera (alguno de los esquemas previos), y los clusters están conectados, por ejemplo, por un *bus*. La comunicación entre cada cluster es local mientras que la comunicación entre clusters es llamada global, esperando que habrá un balance conveniente entre estos tipos de comunicación tal que la mayoría de ella para un procesador sea local y menos frecuente será la necesidad del cluster de comunicarse con otro. Dentro del cluster la conexión puede ser por bus, anillo, malla, hipercubo, entre otros.

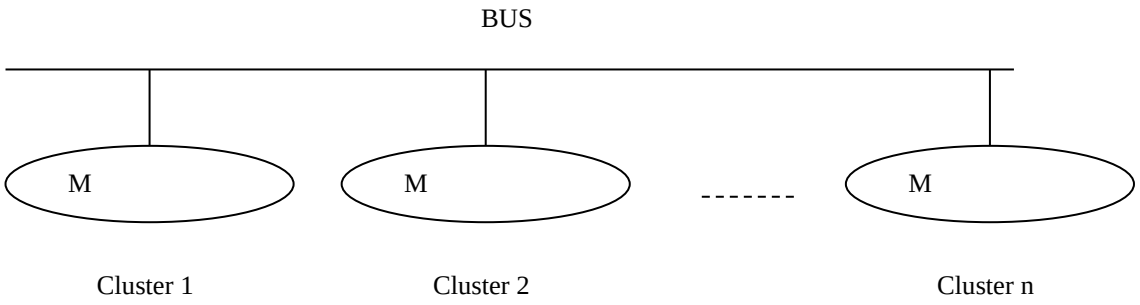


Fig. III.7 Conexión de cluster

En sistemas con memoria distribuida, su comunicación se realiza combinando hardware y software. Los puntos principales en este tipo de comunicación son en el envío de datos que deben ser primero tomados de la memoria del procesador que envía, que debe informarse a donde se enviarán, los datos deben ser físicamente transmitidos entre procesadores y deben colocarse en ubicaciones correctas de memoria del procesador que recibe.

III.3. Modelos de Programación en Paralelo

La organización de la memoria, el control de procesos y el tipo de paralelismo del problema, determinan el modelo de programación a usar, **Cismaşiu (2002)**. Existen varios modelos de uso común: *shared memory* (memoria compartida), *threads* (hebras), *message passing* (paso de mensajes), *data parallel* (datos paralelos), *hybrid* (híbrido), *Single Program Multiple Data* (SPMD), *Multiple Program Multiple Data* (MPMD), etc. que son una abstracción sobre arquitecturas de hardware y memoria, y aunque probablemente no sea algo evidente, estos modelos no son específicos a un tipo de máquina en particular o tipo de arquitectura, por lo que cualquiera de ellos puede (teóricamente) aplicarse a cualquier tipo de hardware.

El modelo que debe usarse es una combinación de la disponibilidad y selección del desarrollador, no hay un modelo que sea el mejor pero si hay implementaciones mejores de unos modelos sobre otros, **Barney (2007)**. A continuación se da una descripción breve de los modelos mencionados:

Memoria Compartida

En este modelo, las tareas comparten un espacio común de direcciones en donde leen y escriben de forma asíncrona, además, para su desarrollo se emplean semáforos o cerraduras para controlar el acceso a dicho espacio. Una ventaja de este modelo desde el punto de vista del programador es que no existe la necesidad de especificar de forma explícita la comunicación de datos entre tareas por lo que el desarrollo puede en ocasiones ser simplificado. La desventaja más importante en términos de comportamiento es su dificultad de entendimiento y el manejo de localidad de datos. Para su implementación, en plataformas de memoria compartida, los compiladores nativos traducen las variables del programa en direcciones actuales de memoria, las cuales son globales.

Hebras

En el modelo de programación de hebras, un solo procesador puede tener múltiples rutas de ejecución concurrentes. Una forma simple de ejemplificar este modelo se presenta cuando un programa durante su ejecución de manera serial llega a un punto en el que crea un número de tareas (hebras) que trabajan de forma concurrente y se comunican entre ellas a través de la memoria global por lo que se requieren de estructuras de sincronización. Las hebras se asocian comúnmente con arquitecturas y sistemas operativos de memoria compartida. La implementación comprende comúnmente: una librería de subrutinas que se llaman desde el código paralelo y un conjunto de directivas del compilador integradas a la parte serial y paralela del código.

Paso de Mensajes

Este modelo se emplea en programas que presentan un conjunto de tareas que usan su propia memoria local durante el cómputo. Múltiples tareas pueden estar en la misma máquina (o en varias) y la transferencia requiere operaciones que cooperen entre los procesadores, por ejemplo, una operación de emisión en un procesador debe tener otra de

recepción en otro procesador. La comunicación en el paso de mensajes se hace por medio de operaciones de envío y recepción empleando un software especializado.

Las librerías de paso de mensajes permiten desarrollar eficientes programas en sistemas de memoria distribuida. Estas librerías proveen rutinas para iniciar y configurar el ambiente de mensajes para enviar y recibir paquetes de datos. Cuando se escriben aplicaciones paralelas, el programador tiene que desarrollar una cantidad importante de software para manejar algunas tareas como: la comunicación y sincronización entre procesos, y la entrada y salida de estructuras de datos.

Datos Paralelos

Este modelo de programación centra la mayoría del trabajo paralelo en realizar operaciones sobre un conjunto de datos el cual se organiza en una estructura. El conjunto de tareas trabaja de manera colectiva en la misma estructura de datos, sin embargo, cada tarea trabaja en una diferente partición de la estructura y realiza la misma operación que las demás, por lo que se requiere que un procesador sea responsable de ensamblar los datos en un camino concurrente; en arquitecturas de memoria compartida, todas las tareas pueden tener acceso a la estructura de datos a través de memoria global, mientras que en arquitecturas de memoria distribuida la estructura de datos es dividida y se guarda en pedazos. La programación con este modelo generalmente se complementa escribiendo un programa con constructoras paralelas de datos. Un programa paralelo con este modelo de programación es una secuencia de instrucciones paralelas explícitas e implícitas.

Híbrido

En este modelo se combinan dos o más modelos de programación, un ejemplo común es la combinación de paso de mensajes con el modelo de hebras o el de memoria compartida.

SPMD

Este es un modelo de alto nivel y puede construirse a través de la combinación de los modelos mencionados, este programa ejecuta las tareas de forma simultánea y en cualquier momento las instrucciones de las tareas pueden ser las mismas o diferentes dentro del mismo programa; generalmente posee programación lógica para permitir a las tareas ejecutar partes del programa en particular.

MPMD

Es otro modelo de programación de alto nivel (como el modelo SPMD), se crea a través de la combinación de los modelos anteriores y tiene múltiples archivos ejecutables (programas) y al ejecutarse en paralelo, cada tarea puede ejecutar los mismos o diferentes programas.

Los modelos SPMD y MPMD se generan a partir de la descomposición en procesos o en tareas elementales del problema de cómputo, siendo necesario implementar explícitamente la comunicación entre procesos.

III.4. Paradigmas de Programación en Paralelo

Las aplicaciones paralelas pueden clasificarse dentro de paradigmas bien definidos donde cada uno es una clase de algoritmos que tienen la misma estructura de control, y se puede decir que hay un número pequeño de éstos detrás de la mayor parte de los programas paralelos. Su elección depende de los recursos disponibles y del tipo de paralelismo inherente al problema, los recursos computacionales pueden definir el grado de granularidad que puede soportar eficientemente el sistema y el tipo de paralelismo refleja la estructura de la aplicación (o de los datos), además, ambos tipos pueden existir en diferentes partes del programa, **Buyya (1999)**. Existen muchos autores que presentan una clasificación de paradigmas de programación en paralelo, sin embargo, los que han sido más usados son los siguientes:

Maestro Esclavo

Este paradigma consta de dos entidades: un maestro y múltiples esclavos. El maestro es el responsable de descomponer el problema en pequeñas tareas y distribuirlas entre los procesos esclavos, también, sirve para almacenar resultados parciales en orden para producir el resultado final. Los procesos esclavos consiguen un mensaje con la tarea, la procesan y envían el resultado al maestro; generalmente, la comunicación se da entre el maestro y los esclavos.

Para su desarrollo se puede usar un balance de carga estático o dinámico. En el primero, la distribución de las tareas se realiza al principio del cómputo, lo que le permite al maestro participar en el cómputo después de que cada esclavo ha alojado una fracción del trabajo. Por otro lado, el balance de carga dinámico aplicado en este paradigma es más benéfico cuando el número de tareas: es mayor que el número de procesadores, se desconoce al comienzo de la aplicación, en sus tiempos de ejecución no son predecibles o cuando el problema está sin balance. Además, el balance de carga dinámico tiene la habilidad de adaptarse a las condiciones de cambio del sistema.

El paradigma de maestro-esclavo puede alcanzar altas velocidades de procesamiento y tener un interesante grado de escalabilidad. Sin embargo, para un gran número de procesadores, el control centralizado del nodo maestro puede volverse un problema de congestionamiento para la aplicación, aunque es posible, mejorar la escalabilidad del paradigma extendiendo el nodo maestro a un conjunto de maestros, donde cada uno de ellos controla un diferente grupo de procesadores esclavos. En la **Fig. III.8** se muestra la representación esquemática de este paradigma.

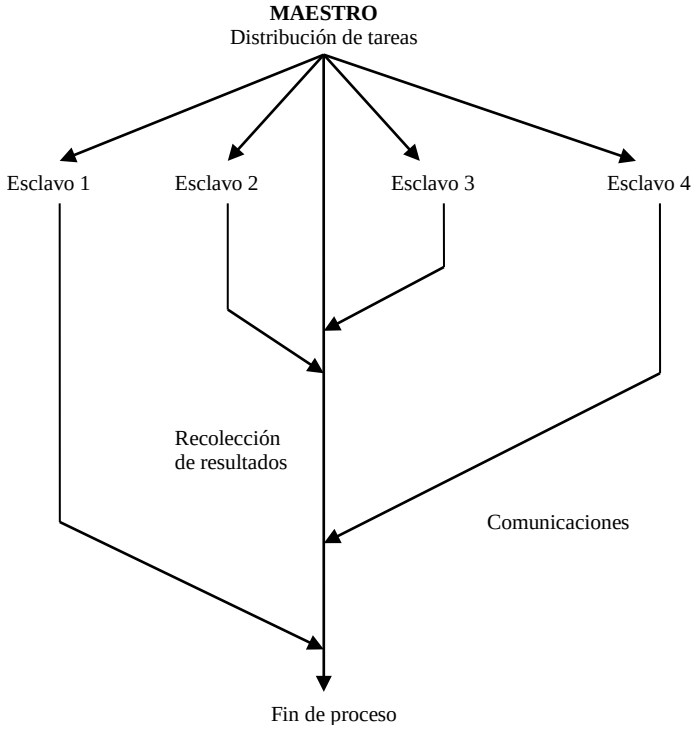


Fig. III.8 Estructura maestro esclavo

SPMD

El paradigma SPMD es el que se usa comúnmente, donde cada proceso ejecuta básicamente la misma parte de código pero en una parte diferente de datos. Esto involucra la división de los datos de la aplicación entre los procesadores. Este tipo de paralelismo también es referido como paralelismo geométrico, descomposición de dominio o paralelismo de datos; lo representamos en la **Fig. III.9**.

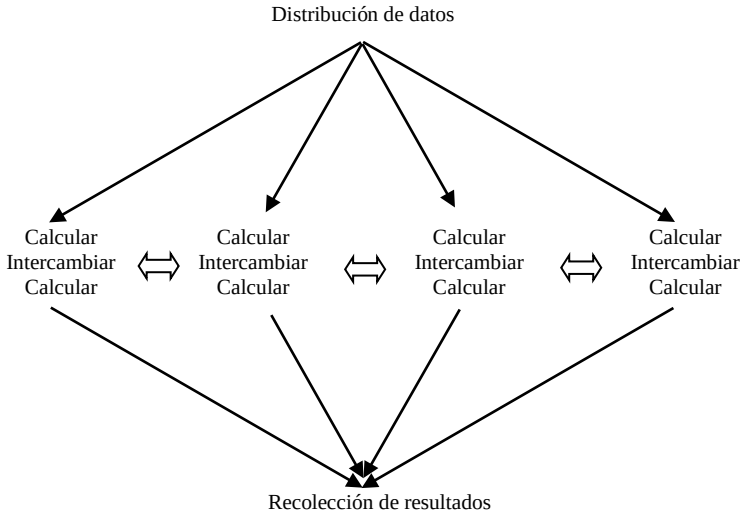


Fig. III.9 Estructura básica de un programa SPMD

Muchos problemas físicos tienen una estructura geométrica regular con interacciones limitadas espacialmente. Esta homogeneidad permite distribuir los datos uniformemente entre los procesadores para que cada uno sea responsable de un área definida del problema. Los procesadores se comunicarán con sus vecinos y la carga de comunicación será proporcional al tamaño de las fronteras del elemento, mientras que la carga de cómputo será proporcional al volumen del elemento, en ocasiones, puede requerirse realizar una sincronización global de forma periódica entre todos los procesos.

Las aplicaciones SPMD pueden ser muy eficientes si los procesos distribuyen bien los datos y el sistema es homogéneo. Si los procesos presentan diferentes cargas de trabajo o capacidades, el paradigma requerirá de un esquema de balance de carga capaz de adaptar la distribución de datos durante el tiempo de ejecución.

Este paradigma es altamente sensitivo a la pérdida de algunos procesos, generalmente, la pérdida de un proceso es suficiente para ocasionar un punto muerto en los cálculos, donde, ninguno de los procesos puede avanzar más allá del punto de sincronización global.

Data Pipelining

Este es el paradigma de paralelismo con la granularidad más fina que se basa en una descomposición funcional, para este caso, en las tares del algoritmo. Las tareas que son capaces de operar de forma concurrente son identificadas y cada procesador ejecuta una pequeña parte del algoritmo total. El *pipeline* (o línea de trabajo) es uno de los paradigmas de descomposición funcional más simples y populares. En la **Fig. III.10** se presenta la estructura de este modelo.

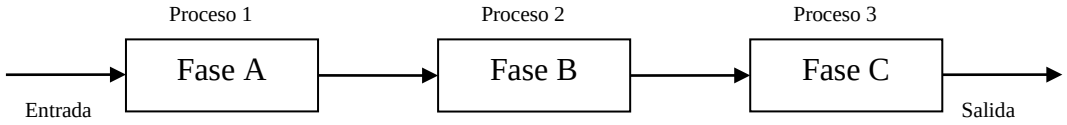


Fig. III.10 Estructura de línea de datos

Los procesos se organizan en una línea de trabajo donde cada uno de ellos corresponde a una etapa y es responsable de una tarea en particular. El patrón de comunicación puede ser muy simple ya que el flujo de datos se realiza entre las etapas adyacentes de la línea, por esta razón, este tipo de paralelismo en ocasiones es referido como paralelismo de flujo de datos y la comunicación puede ser totalmente asíncrona. La eficiencia de este paradigma depende de la habilidad para equilibrar la carga a través de las etapas de la línea de trabajo. La robustez de este paradigma contra la reconfiguración del sistema, puede alcanzarse considerando múltiples rutas independientes entre las etapas. Este paradigma en ocasiones se usa en la reducción de datos o en el procesamiento de imágenes.

Dividir y Conquistar

El paradigma de dividir y conquistar es bien conocido en el desarrollo de algoritmos secuenciales; bajo éste, se divide un problema en dos o más subproblemas, donde cada uno de ellos es resuelto independientemente y sus resultados son combinados para dar una solución final. En ocasiones, los problemas más pequeños son instancias menores del problema original, lo que da lugar a una solución recursiva. El procesamiento puede ser requerido para dividir el problema original o para combinar los resultados del subproblema. En paralelo, los subproblemas pueden ser resueltos al mismo tiempo, y los procesos de combinación y división requieren de comunicación, sin embargo, si los subproblemas son independientes no es necesaria.

Podemos identificar tres operaciones computacionales genéricas en este paradigma: dividir, calcular y juntar. La aplicación se organiza en tipo de árbol virtual: algunos de los procesos crean subtareas y tienen que combinar los resultados de ellas para producir uno final, esto se puede observar en la **Fig. III.11**.

El paradigma maestro esclavo puede verse como una modificación de dividir y conquistar, donde por ejemplo, la descomposición del problema se realiza antes de enviar a las tareas, las operaciones de dividir y juntar se hacen por medio del proceso maestro y los demás son responsables únicamente del cómputo.

Los paradigmas de programación pueden ser caracterizados principalmente por dos factores: descomposición y distribución del paralelismo. En paralelismo geométrico, la descomposición y distribución son estáticas; lo mismo sucede con la descomposición funcional y la distribución de línea de datos; en el modelo de maestro esclavo, el trabajo es descompuesto de forma estática y distribuido dinámicamente; finalmente, en el paradigma de dividir y conquistar la descomposición y división son dinámicas.

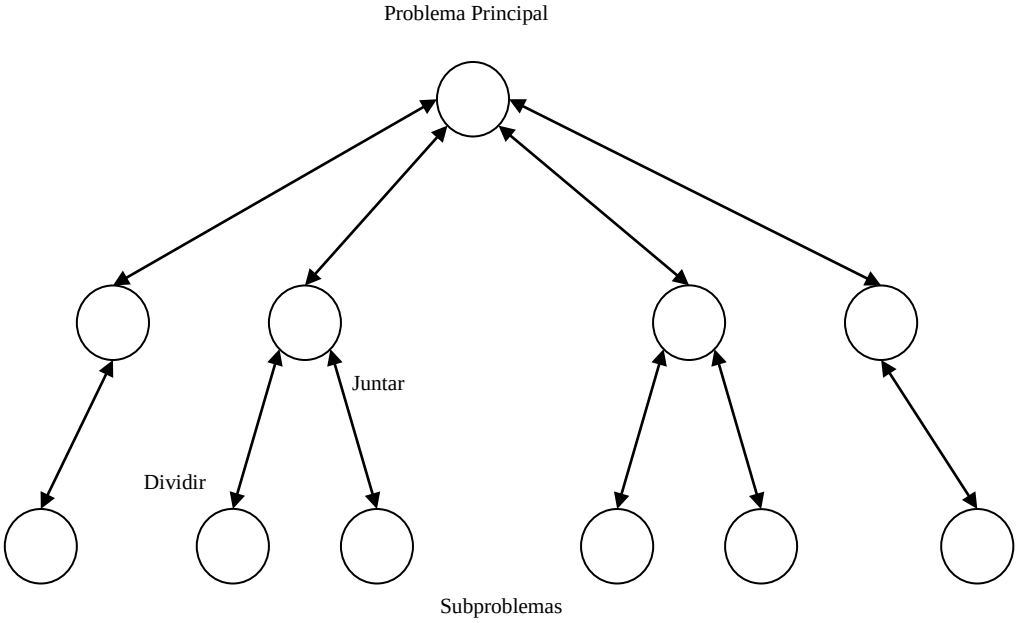


Fig. III.11 Paradigma dividir y conquistar como árbol virtual

Paralelismo Especulativo

Este paradigma se usa cuando es difícil obtener paralelismo por medio de alguno de los anteriores. Algunos problemas tienen complejas dependencias de datos, las cuales reducen las posibilidades de explotar la ejecución en paralelo. En estos casos, una solución apropiada es ejecutar el problema en partes pequeñas, pero emplear un poco de especulación o ejecución optimista para facilitar el paralelismo. En ocasiones, en lugar de intentar determinar las dependencias entre las iteraciones, se ejecuta en paralelo suponiendo que no hay dependencias, “ejecución optimista", en donde un mecanismo de control (hardware o software) debe encargarse de detectar violaciones de dependencia en tiempo de ejecución y descartar los valores calculados incorrectamente, de esta forma, para cada variable compartida se comprueba si alguno de los hilos que se encuentra ejecutando iteraciones posteriores ha utilizado un valor antiguo de esa variable.

Otro uso de este paradigma consiste en emplear diferentes algoritmos para el mismo problema, y el primero en dar la solución final será el elegido.

Paradigmas Híbridos

Las fronteras entre los paradigmas pueden ser difusas en ocasiones, y en algunas aplicaciones, puede haber la necesidad de mezclar elementos de diferentes de ellos. Los métodos híbridos que incluyen más de un paradigma básico son generalmente observados en aplicaciones paralelas de gran escala. Éstas son situaciones donde tiene sentido mezclar paralelismo de datos y tareas de forma simultánea, o emplearlos en diferentes partes de un mismo programa.

III.5. Diseño de Programas en Paralelo

No existe una receta para diseñar algoritmos paralelos, sin embargo, una metodología de diseño permite al programador centrarse en diversos temas para realizarlo. Los algoritmos paralelos realizan cálculos de forma simultánea con el objetivo de reducir el tiempo de cómputo e incrementar la disponibilidad de memoria. Para alcanzar esto, el esfuerzo computacional debe ser dividido en una colección de tareas independientes que se pueden ejecutar en paralelo interactuando entre ellas.

El diseño de algoritmos en paralelo debe enfrentar varios problemas complejos incluyendo manipulación de datos, asignación de memoria, y además debe poseer concurrencia. Para realizar dicho algoritmo se recomienda un desarrollo metódico que estructure el proceso de diseño en cuatro etapas: partición, comunicación e interferencia de tareas, aglomeración y mapeo; estas etapas fueron propuestas por **Foster (1995)** en donde las dos primeras buscan desarrollar algoritmos concurrentes y escalables, mientras que las dos últimas se centran en aspectos relacionados en localidad y comportamiento. En la **Fig. III.12** mostramos de forma esquemática estas etapas de diseño.

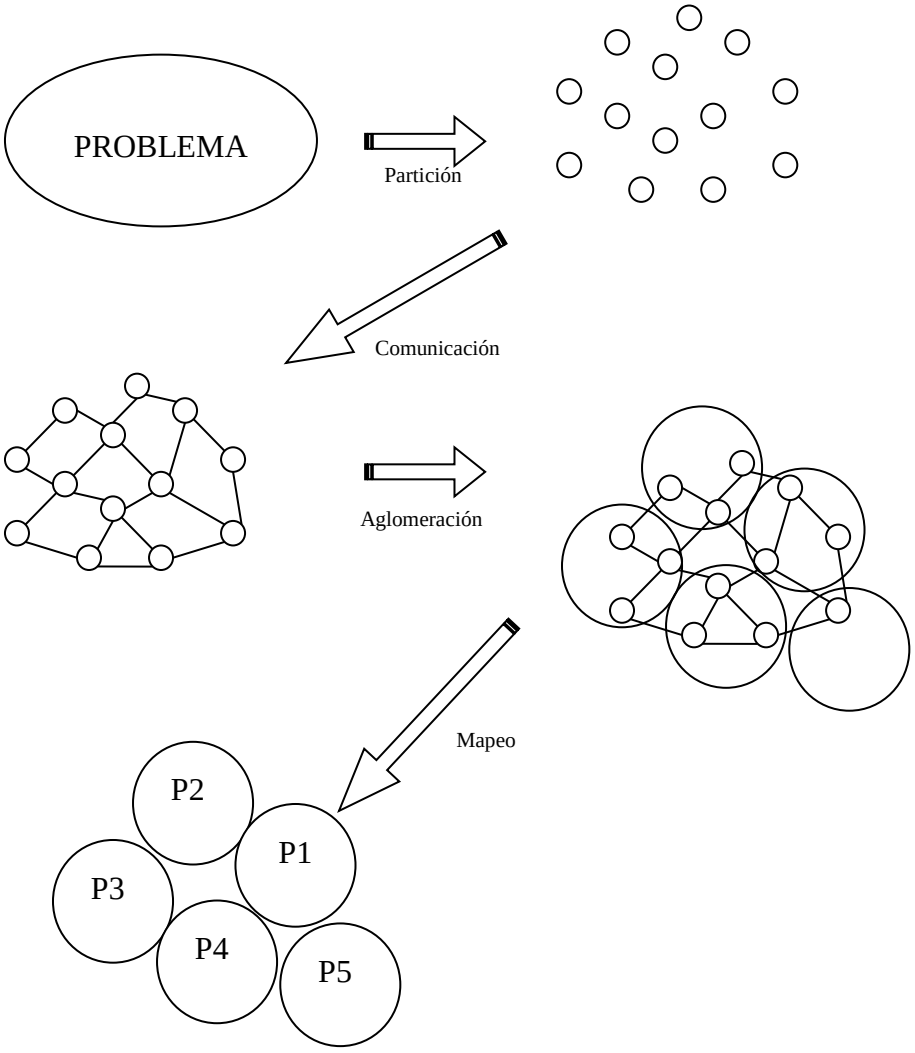


Fig. III.12 Etapas de desarrollo de algoritmos paralelos

III.5.1. Partición

Esta etapa implica la descomposición del problema computacional y de los datos en un conjunto de tareas concurrentes con la finalidad de exponer áreas de oportunidad para ejecución paralela. Una buena descomposición debe llevarnos a tener un alto grado de concurrencia y poca interferencia entre las tareas. Sin embargo, existe un número máximo de tareas que pueden ejecutarse simultáneamente en un programa paralelo, el cual define el grado de concurrencia del algoritmo y es una medida de qué tan efectiva puede ser ejecutada una aplicación en paralelo.

Varias de las técnicas usadas en la descomposición de un problema son dependientes exclusivamente del modelo en estudio. Para una aplicación de elemento finito, por ejemplo, se pueden tomar varias estrategias: descomposición de dominio, subestructuración, división de operadores y elemento por elemento.

Descomposición de Dominio. Esta técnica ha sido empleada para resolver problemas de gran escala dividiéndolo en piezas pequeñas. En el caso de elemento finito, la malla se divide en subdominios y cada uno de ellos posee un número de elementos.

Subestructuración. Esta técnica se relaciona con la descomposición de dominio. Puede ser identificada a nivel algebraico dividiendo las matrices asociadas.

División de Operadores. Esta división puede ser usada para seccionar las tareas de cómputo en subtareas. Ellas son independientes o interactúan poco, por lo que el cómputo puede realizarse en diferentes procesadores con un poco de comunicación o compartiendo información.

Elemento por elemento. Esta estrategia se usa para desarrollar soluciones en análisis de elemento finito que no requiere la generación explícita de la matriz global.

III.5.2. Comunicación e Interferencia de Tareas

Las tareas generadas por una partición se ejecutan de forma concurrente, pero en general, no pueden ejecutarse independientemente, por lo que requieren cooperar entre ellas. En esta etapa se determinan las comunicaciones requeridas para coordinar la ejecución de las tareas, se especifica el flujo de la información y se definen estructuras de comunicación apropiadas con sus algoritmos, además, el diseño debe hacerse considerando la arquitectura y el modelo de programación seleccionados.

Cuando se usa descomposición de dominio, los requerimientos de comunicación pueden aparecer en varios momentos, ya sea para manejar la transferencia de datos necesaria para iniciar las tareas y para obtener la solución global del problema. Los patrones de comunicación pueden ser clasificados como: local/global, estructurado/no estructurado, estático/dinámico, síncrono/asíncrono.

En la comunicación local, cada tarea se comunica con un conjunto pequeño de tareas, mientras que en la global cada tarea se comunica con varias. En la comunicación estructurada, una tarea y sus tareas vecinas forman una estructura regular; en la no estructurada se tiene una red de forma arbitraria. Para el patrón estático, la identidad de las parejas de comunicación no cambia con el tiempo, pero en el dinámico, dicha identidad puede determinarse por el cómputo de datos durante la ejecución, pudiendo ser altamente variable. En la comunicación síncrona, emisores y receptores se ejecutan de forma coordinada cooperando en las operaciones de transferencia de datos, en el patrón asíncrono, el receptor requiere datos del emisor.

III.5.3. Aglomeración

Las tareas y estructuras de comunicación definidas en las dos primeras etapas de diseño, se evaluan en términos de costo de comunicación. Si es necesario, las tareas se combinan para mejorar el comportamiento. Cuando desarrollamos un algoritmo paralelo o se paraleliza código existente hay peligro de convertir el problema computacional en un problema de comunicación. Esto puede ocurrir cuando el problema es dividido en forma

excesiva, tal que, el tiempo empleado para comunicar datos entre nodos y para sincronizar rebasa el tiempo de cómputo actual y la etapa de aglomeración puede eliminarlo. Puede ser necesario examinar las decisiones tomadas en las fases de partición y comunicación para obtener un algoritmo que se ejecute eficientemente en la computadora paralela. En ocasiones, es útil combinar o aglomerar, tareas identificadas durante la partición para obtener un pequeño número de ellas, de gran tamaño o para determinar dónde vale la pena replicar datos o efectuar nuevamente los cálculos. Esta fase de diseño puede ser interpretada como el proceso de ajuste del algoritmo a la arquitectura de la computadora, y los resultados pueden expresarse por la granularidad del algoritmo.

El grado de granularidad de un algoritmo depende de que tan eficientemente el hardware soporte la comunicación entre procesadores. Si la comunicación es rápida y su costo se compara con el costo computacional, algoritmos con granularidad fina pueden emplearse. Si por el contrario es costosa, algoritmos con grandes relaciones cómputo/comunicación se recomiendan, obteniendo paralelismo de granularidad gruesa. Paralelización de granularidad media se lleva a cabo si grupos de elementos adyacentes se asignan a un procesador. Existe un grado mínimo de granularidad, bajo el cual la sobrecarga de comunicación excede el costo computacional y no brinda ventajas por distribuir el problema.

III.5.4. Mapeo

La etapa final en el diseño de algoritmos paralelos se refiere a la ubicación en donde cada una de las tareas se ejecutará, con el objetivo de reducir el tiempo de ejecución al tener un conjunto de tareas y uno de procesadores, varios mapeos son posibles, pero no todos son los óptimos. Un mapeo óptimo debe ser aquel que minimice los costos de comunicación y maximice el uso del procesador. Las decisiones de mapeo pueden ser estáticas (antes de la ejecución del programa) o dinámicas (durante el tiempo de ejecución por medio de técnicas de balance de carga.

Muchos algoritmos desarrollados usando descomposición de dominios contienen un número fijo de tareas de igual tamaño y comunicaciones estructuradas locales y globales, en estos casos, un mapeo eficiente es sencillo y se puede usar un balance de carga estático. En problemas más complejos donde el número de tareas, la cantidad de cómputo o la comunicación por tarea varían dinámicamente durante la ejecución del programa, una estrategia de balance de carga dinámico es bastante útil para determinar una nueva aglomeración y mapeo. Para reducir la sobrecarga de comunicación las alternativas posibles son: reducir la interacción maximizando la localidad de datos, sustituir la comunicación con cálculos computacionales y replicar datos u operaciones de cómputo.

Las cuatro etapas presentadas son necesarias para el diseño de algoritmos paralelos, pero no pueden capturar todos los aspectos relacionados a este proceso como la calidad del algoritmo, costos de implementación, portabilidad, rehúso de código, modularidad, etc. El objetivo en la programación en paralelo es optimizar varias métricas como tiempo de ejecución, requerimientos de memoria y costos de implementación.

Los elementos empleados como métricas para evaluar el comportamiento de los programas paralelos son el tiempo de ejecución, eficiencia en paralelo, requerimientos de memoria, rendimiento, latencia, costos de diseño e implementación, requerimientos de hardware, portabilidad y escalabilidad, la importancia relativa de cada una de ellas depende de la naturaleza del problema.

III.6. MPI (Message Passing Interface)

Las librerías de MPI, son un estándar portátil de paso de mensajes que facilita el desarrollo de aplicaciones en paralelo que define sintaxis y semántica; dan portabilidad a los programas en paralelo por lo que se tiene la posibilidad de ejecutar el mismo código en diferentes computadoras (siempre y cuando estén implementadas las librerías) sin necesidad de modificar el código fuente. **Zhuang (1995)**, establece lo siguiente: "*MPI es un conjunto de especificaciones para el envío de mensajes elaboradas por un conjunto de universidades, laboratorios, desarrolladores de software y manufactureras de hardware*", estas especificaciones se emplean principalmente en computadoras paralelas con memoria distribuida, en las que el intercambio de datos se realiza por medio de librerías de comunicación personalizadas para cada tipo de computadora. En las arquitecturas pertenecientes al modelo MIMD, las librerías de paso de mensajes son las más usadas por su portabilidad.

MPI da al programador una colección de funciones para que éste diseñe su aplicación, sin que tenga que conocer necesariamente el hardware concreto sobre el que se va a ejecutar, ni la forma en la que se han implementado las funciones que emplea; se usa para especificar la comunicación entre un conjunto de procesadores que forman un programa concurrente. El paradigma de paso de mensajes es atractivo por su amplia portabilidad y escalabilidad, siendo compatible con memoria distribuida, compartida y con combinaciones de ellas. Como lo menciona **Dongarra y cols. (1995)** y **Alonso (1997)**, MPI ha definido características importantes tales como tipos de datos definidos por el usuario, puertos de comunicación persistentes, operaciones de comunicación colectiva de gran alcance y mecanismos para comunicación. Las librerías tienen como objetivos:

- Diseño de una aplicación portátil.
- Permitir comunicaciones altamente eficientes en varias plataformas.
- Permitir la implementación para sistemas homogéneos.
- Permitir la mezcla de lenguajes como C++ y Fortran.
- Proveer una interfaz que sea consistente con una amplia variedad de hardware y sistemas operativos.
- Proveer una interfaz de programación que no requiera que el programador se ocupe de las fallas de comunicación.
- Permitir implementaciones que puedan proveer múltiples hebras de ejecución dentro de cada proceso.

Las aplicaciones paralelas que emplean el modelo de envío de mensajes son no deterministas ya que el orden de llegada de mensajes de dos procesos A y B, a un tercero C no está definido, siendo responsabilidad del programa asegurar que el cómputo sea determinista cuando así se requiera.

El estándar (en su primera versión) no soporta algunos aspectos como operaciones de memoria compartida; mensajes de interrupción, ejecución remota y mensajes activos; herramientas de construcción de programas; soporte para eliminar errores; soporte de hebras o hilos de ejecución; manejo de procesos o tareas; funciones de entrada y salida. La razón principal de no contemplar estas restricciones fue tiempo y el sentimiento de que son dependientes del sistema.

III.6.1. Especificaciones de MPI

MPI especifica comunicaciones punto a punto (mensajes entre pares de procesadores), comunicaciones colectivas, comunicadores, grupos de procesos, topologías de procesos, vínculos para C y Fortran, interfaz de perfilamento, administración del ambiente y funciones de indagación; de estas especificaciones se muestran algunos aspectos relevantes.

III.6.1.1. Comunicaciones Punto a Punto

Entre otras cosas, un comunicador sirve para definir el conjunto de procesos permitidos dentro de una operación de comunicación. Para la comunicación punto a punto, MPI da un conjunto de funciones para enviar y recibir información, permitiendo la comunicación de tipos de datos con una etiqueta asociada, además, define dos modelos de comunicación: con bloqueo y sin bloqueo, los cuales tienen que ver con el tiempo que un procesador pasa bloqueado tras llamar a una función de comunicación. Una función con bloqueo mantiene detenido a un procesador hasta que la operación solicitada finalice, mientras, una función sin bloqueo delega al sistema la realización de una operación, recuperando el control inmediatamente y más adelante tiene que averiguar si finalizó o no dicha operación.

Independientemente de si la función invocada tiene bloqueo o no, el programador tiene un cierto control sobre la forma en que se realiza y completa un envío. Con base en esto, en la comunicación punto a punto existen modos de comunicación que permiten elegir la semántica de la operación de envío e influye en el protocolo de transferencia de datos, los cuales se detallan a continuación:

Modo Estándar. En este modo, la terminación del envío no significa necesariamente que la recepción ha comenzado y no se deben hacer suposiciones en el programa si los datos están almacenados en MPI, es decir, no se especifica la forma en que se completa la operación por lo que es algo dependiente de la implementación.

Modo con Buffer. Es donde el usuario puede garantizar que cierta cantidad de buffer está disponible, la cual debe ser dada explícitamente por el programa. Cuando se hace un envío con buffer se guarda inmediatamente una copia del mensaje y la operación se da por

completa en cuanto se ha efectuado esta copia; si no hay espacio en el buffer, el envío fracasa.

Modo Síncrono. Aquí se usa la semántica entre el emisor y receptor. Si se hace un envío en este modo, la operación se da por terminada solo cuando el mensaje ha sido recibido en su destino.

Modo Ready. Este modo permite al usuario explotar conocimiento extra para simplificar el protocolo y potencialmente alcanza mayor comportamiento, en el envío el usuario afirma que la recepción ya está fijada, esto es, sólo se puede hacer si antes el otro extremo está preparado para una recepción inmediata. No hay copias adicionales del mensaje, y tampoco podemos confiar en bloquear hasta que el receptor esté preparado.

Todas las funciones de comunicación en MPI toman un tipo de dato como argumento, en el caso más simple es un tipo primitivo que puede ser un dato entero o real. Una importante y poderosa generalización resulta de permitir tipos definidos por el usuario donde los tipos primitivos pueden aparecer. Estos no son tipos como los que posee el propio lenguaje, son tan sólo tipos de los que MPI está enterado por medio del uso de funciones constructoras y describen la disposición en memoria, de conjuntos de tipos primitivos, con ellos se soporta la comunicación de estructuras de datos complejas tales como secciones de arreglos y estructuras que contienen combinaciones de tipos de datos primitivos.

Las comunicaciones punto a punto son útiles en arquitecturas o paradigmas de programación de maestro-esclavo, donde el nodo maestro debe ser responsable del manejo de flujo de datos en una colección de nodos esclavos. Típicamente el nodo maestro enviará grupos específicos de instrucciones o datos a cada nodo esclavo, y comparará los resultados al término. Las funciones de recepción de mensajes engloban en una operación la sincronización del emisor (esperar a que haya un mensaje disponible) con la comunicación (copiar ese mensaje), aunque en ocasiones conviene separar ambos conceptos.

III.6.1.2. Comunicaciones Colectivas

Muchas aplicaciones requieren de la comunicación entre más de dos procesadores. Esto se puede hacer combinando comunicaciones punto a punto, pero para que resulte más cómodo al programador, y para permitir implementaciones optimizadas, MPI incluye comunicaciones colectivas, las que transmiten datos entre todos los procesadores especificados por un objeto comunicador. Brevemente, MPI provee las siguientes funciones de comunicación colectiva:

- Barrera de sincronización a través de todos los procesos.
- Difusión de un proceso a todos.
- Recolección de datos de todos en uno.
- Distribución de datos de uno a todos.
- Recolección y dispersión de datos.

- Reducción de operaciones globales como suma, máximo, mínimo y funciones definidas por el usuario.
- Exploración a través de los procesos.
- Combinaciones de todas ellas.

Una operación de comunicación colectiva tiene que ser invocada por todos los participantes, aunque los roles que jueguen no sean los mismos. La mayor parte de estas comunicaciones requieren la designación de un proceso como raíz de la operación.

La sintaxis y semántica de las funciones colectivas de MPI fueron diseñadas para ser consistentes con comunicaciones punto a punto. Sin embargo para mantener el número de funciones y su lista de argumentos a un nivel razonable de complejidad, se crearon funciones más restrictivas que las funciones punto a punto en varios caminos, una restricción es que la cantidad de datos enviados debe ser exactamente la misma que la de los datos recibidos.

Las funciones de barrera de sincronización y de difusión son las más comunes; la barrera no exige ninguna clase de intercambio de información y sirve para sincronizar un grupo de procesos sin transmisión de datos, bloquea a los procesos de un comunicador hasta que todos ellos han pasado por ella y suele emplearse para dar por finalizada una etapa del programa, asegurándose de que todos han terminado antes de dar comienzo a la siguiente; las funciones de difusión sirven para que un proceso, el raíz, envíe un mensaje a todos los miembros del comunicador.

Además de las funciones mencionadas, tenemos las de recolección que realizan una recopilación de datos en el proceso raíz en un vector de datos, donde se almacenan de forma consecutiva las contribuciones de todos los procesos y cuando se requiere difundir los resultados de la recolección, se concatena a ella una función de difusión.

Otro tipo de funciones son las de distribución donde el proceso raíz posee un vector de elementos por cada proceso del comunicador y después de realizar la función cada proceso tiene una copia del vector inicial.

Finalmente las reducciones, dentro de las funciones colectivas, son operaciones realizadas de forma cooperativa entre todos los procesos de un comunicador, de tal forma que se obtiene un resultado final que se almacena en el proceso raíz. MPI define una colección de funciones que se pueden utilizar en una reducción, aunque, el programador puede necesitar una función distinta, no predefinida, por lo que se ofrecen los medios para crearla. En ocasiones es útil que el resultado de una reducción esté disponible para todos los procesos por lo que se puede combinar con una función de difusión o una de distribución.

III.6.1.3. Comunicadores

Una característica clave necesitada para soportar la creación de robustas librerías paralelas, es garantizar que la comunicación dentro de una rutina no entre en conflicto con

comunicaciones extrañas a ella. El concepto encapsulado por un comunicador de MPI da este soporte.

Un comunicador es un objeto que especifica el alcance de una operación de comunicación, que puede ser el grupo de procesos involucrados y el contexto de comunicación el cual parte el espacio de comunicación. Un mensaje enviado en un contexto no puede ser recibido en otro contexto y la fila de procesos es interpretada con respecto al grupo de procesos asociado con un comunicador. Estos comunicadores son especialmente importantes para el diseño de librerías de software paralelo. Un comunicador provee un mecanismo conveniente para el paso del grupo de procesos involucrados dentro de las librerías y de la rutina, la fila de procesos será interpretada relativa a este grupo.

III.6.1.4. Grupos y Topologías de Procesos

Un grupo es una colección ordenada de procesos y define un espacio de direcciones, esto es, los miembros del grupo tienen asignada una dirección dentro de él. Un proceso puede pertenecer simultáneamente a varios grupos y tener una dirección distinta en cada uno de ellos. En estos grupos se dice como se usan y manipulan los procesos.

La topología de procesos se refiere a las formas físicas de integrar y distribuir los procesos. Se refiere a funciones que permiten la manipulación conveniente de etiquetas de procesos cuando los procesos son considerados parte de una topología específica. Una topología provee un mecanismo de nombramiento de procesos conveniente al grupo, y puede asistir al sistema durante la ejecución para el mapeo de procesos dentro del hardware.

III.6.1.5. Vínculos para C y Fortran e Interfaz de Perfilamiento

MPI fue diseñado de tal forma que las versiones para C y Fortran posean una sintaxis sencilla, de hecho, la forma detallada de la interfaz en estos dos lenguajes se especifica y es parte del estándar. En el caso de C, los nombres de las funciones son como en la definición de MPI pero con el prefijo MPI y la primera letra del nombre de la función con mayúscula. Los valores de estado se regresan como códigos de tipo entero. En Fortran, los nombres de las funciones se escriben en mayúsculas, los códigos de regreso se representan por un argumento adicional de tipo entero.

La interfaz se diseña de forma que el perfilamiento de la ejecución o monitoreo del comportamiento puede sumarse al sistema de paso de mensajes. No es necesario tener acceso al código de MPI para hacer esto y, por lo tanto, sistemas portátiles de perfilamiento pueden ser fácilmente construidos.

III.6.2. Tipos Derivados de Datos

MPI maneja en todas sus funciones de envío y recepción, vectores de tipos simples. En general, se asume que los elementos de esos vectores están almacenados consecutivamente en memoria, sin embargo, en ocasiones es necesario el intercambio de tipos estructurados o de vectores no almacenados de forma consecutiva. Estas librerías, incluyen la posibilidad de definir tipos más complejos empleando constructores y permiten definir tipos homogéneos y heterogéneos de datos, en el primer tipo, todos los elementos constituyentes son del mismo tipo mientras que en el segundo, no lo son.

Estos tipos derivados nos permiten eliminar las operaciones de copia de datos ya que si no contáramos con ellos, el envío de un arreglo de dos dimensiones almacenado por columnas requeriría que los elementos no contiguos se copiaran a un buffer antes de realizar el envío.

IV. SIMULADOR DE YACIMIENTOS EN PARALELO

Un simulador de yacimientos, como se mencionó en capítulos anteriores, está constituido por un sistema de ecuaciones, obtenidas de la discretización en tiempo y espacio de su modelo matemático. El sistema de ecuaciones es no lineal y se resuelve de forma iterativa aplicando el método de Newton-Raphson para su linealización, en cada iteración el sistema es resuelto y se verifica la convergencia de la solución a partir de las tolerancias de las variables establecidas previamente durante el modelado; cuando converge la solución, se calculan los valores correspondientes a ese paso de tiempo y se realizan los cálculos para el paso siguiente, cuando no converge se reduce el incremento en tiempo y vuelve a iterar.

De manera general, el proceso de simulación numérica de yacimientos comienza con la introducción de la información, ésta incluye parámetros geológicos, hidrológicos y relaciones constitutivas del medio poroso, tales como permeabilidad absoluta y relativa, porosidad, presión capilar, propiedades termo físicas de la roca y de los fluidos, así como las condiciones iniciales y de frontera del sistema; adicionalmente, se requiere información de la malla de simulación y una serie de opciones numéricas y de control de paso de tiempo. A partir de los datos mencionados, el proceso continúa con la construcción de la malla donde se divide al yacimiento en un conjunto de bloques de los cuales conocemos su ubicación y dimensiones. Una vez que se crea la malla de simulación se inicializan las variables primarias con la información de presión capilar y datos PVT, posteriormente, se calculan todas las propiedades de los fluidos en cada celda a sus condiciones de presión y temperatura. Después de la inicialización, se entra en un ciclo de tiempo donde se crea un sistema de ecuaciones con la información que se tiene de la malla y de los pozos, este sistema se resuelve y con la solución volvemos a calcular las propiedades de los fluidos con lo que podemos avanzar gradualmente en el tiempo.

Uno de los objetivos de la simulación de yacimientos es el poder representar de manera confiable el comportamiento de los fluidos dentro del medio poroso para su predicción, esto se logra con mayores cantidades de información de tipo diferente, con la cual, podemos caracterizar mejor las áreas de estudio y si empleamos mallas de pequeña escala logramos modelar con detalle las heterogeneidades presentes en las formaciones almacenadoras de hidrocarburos, de hecho, se han realizado trabajos para poder representar los yacimientos con millones de nodos o celdas, sin embargo, su solución puede obtenerse mediante el empleo de máquinas paralelas y técnicas de programación apropiadas a este tipo de máquinas. Las técnicas de programación en paralelo han logrado superar las limitaciones del tamaño del problema y de resolución del yacimiento presentadas en máquinas de un solo procesador. En una simulación a gran escala se necesita bastante información lo que provoca que tales requerimientos de memoria sean distribuidos entre los procesadores de una máquina paralela, además, el trabajo de programación en paralelo se ha centrado, en la mayoría de los casos, en la paralelización de las rutinas empleadas para resolver el sistema lineal de ecuaciones ya que ahí es donde se concentra el esfuerzo de cómputo, sin embargo, existen otras partes dentro de un simulador que requieren atención y poseen un alto grado de paralelización como lo es la entrada de datos de la malla y de pozos, construcción de la matriz jacobiana, cálculo de coeficientes, inicialización de la malla, control de pozos, cálculo de propiedades de fluidos, etc.

Para la programación en paralelo existen varias herramientas entre las que encontramos las librerías de MPI, que son un estándar de procedimientos para el paso de mensajes que permite la transferencia de datos de un procesador a otro y que en este tipo de problemas han demostrado ser de gran utilidad al emplear como paradigma el paralelismo de datos, donde los cálculos se aplican sobre diferentes conjuntos de datos (modelo *SPMD*).

En el presente capítulo se presentan algunos aspectos importantes de la simulación de yacimientos naturalmente fracturados, el diseño del simulador en paralelo con su algoritmo y las bases de la descomposición de dominio.

IV.1. Consideraciones de Simulación de Yacimientos Naturalmente Fracturados

Antes de presentar el desarrollo del programa en paralelo, debemos de tomar en cuenta una serie de consideraciones en la simulación de yacimientos naturalmente fracturados, la mayoría de los datos requeridos para evaluar la transmisibilidad de la matriz son similares a los que se usan en otras simulaciones y pueden obtenerse de un análisis de núcleo, **Gilman y Kazemi (1983)**, pero, para describir a las fracturas se requieren datos similares cuya medición ha sido descrita por varios autores; las curvas de permeabilidad relativa son unas de las propiedades que se miden para el sistema de fracturas y son difíciles de medir de forma aproximada, ambas fases deben ser móviles para todo el rango de saturaciones, en la **Fig. IV.1** se muestra el comportamiento de estas curvas para el sistema de matriz y de fracturas.

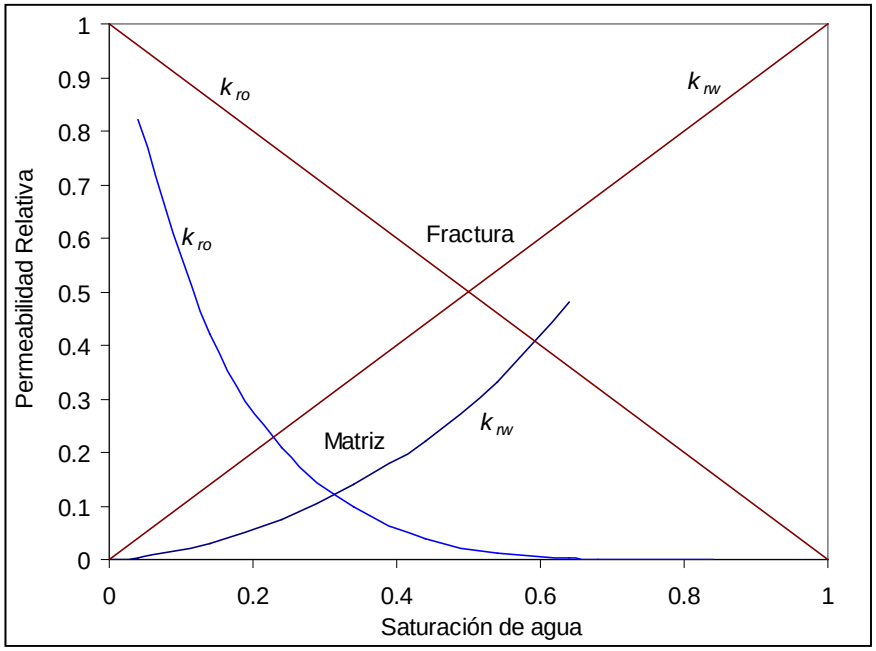


Fig. IV.1 Curvas de permeabilidad relativa agua-aceite para la fractura y matriz

La presión capilar de las fracturas también es otro parámetro difícil de medir, la presión capilar debe ser debe ser cercana a cero para la mayoría de las mediciones de

saturación de agua y, para mantener equilibrio capilar y gravitacional, la presión capilar máxima en la matriz y la fractura deben ser iguales. La diferencia entre las presiones capilares de fractura y matriz representa la fuerza de imbibición, que en ocasiones, es el mayor mecanismo de recuperación en yacimientos fracturados. Las curvas de imbibición son requeridas para los cálculos de recuperación de hidrocarburos, las curvas de drene son empleadas para la inicialización, en la **Fig. IV.2** observamos las curvas de presión capilar para el sistema de matriz y fracturas.

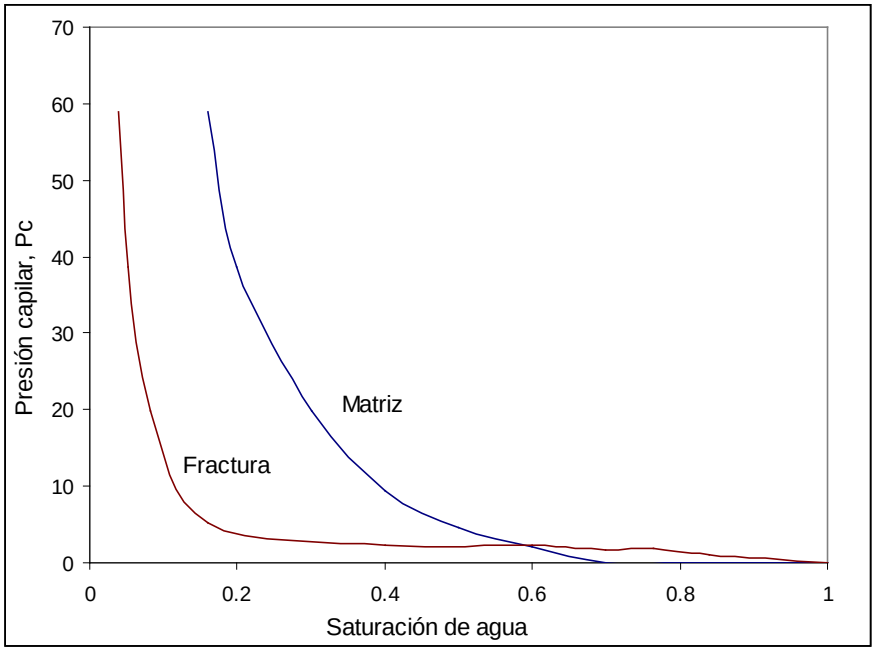


Fig. IV.2 Curvas de presión capilar para la fractura y matriz

En sistemas naturalmente fracturados, la permeabilidad de la fractura correcta para usar es,

$$k_f = \frac{k_e}{\varphi_f} , \tag{4.1}$$

Donde k_e es la permeabilidad efectiva medida por la pendiente de una curva de incremento de presión. La porosidad de la fractura, se define como su volumen dividido por el volumen total,

$$\varphi_f = \frac{V_f}{V_t} , \tag{4.2}$$

El volumen total es el volumen de roca sin fracturas más el volumen de fracturas. La porosidad de la matriz también se define con respecto al volumen total del yacimiento. Por lo tanto, la porosidad de la matriz no es la misma que la porosidad medida de un núcleo en

el laboratorio, la información obtenida en laboratorio, en conjunto con los registros geofísicos, permiten determinarla su valor para cada sistema, matriz y fractura.

Gilman y Kazemi (1983), establecen que el factor de forma, σ, para las fracturas en tres dimensiones, calculado por medio de la ecuación (4.3), nos da el tamaño aparente del bloque de matriz; para flujo en una o dos dimensiones, los correspondientes términos *L* son eliminados. Otros autores han presentado diferentes ecuaciones para σ, si existe una fractura a lo largo de un lado del bloque de matriz para sistemas unidimensionales, el coeficiente en la ecuación (4.3) es 2.0. Para modelos a escala de campo, el factor de forma es una propiedad del sistema y para medios fracturados uniformemente debe ser independiente del tamaño del bloque.

$$\sigma = 4\left(\frac{1}{l_x^2}+\frac{1}{l_y^2}+\frac{1}{l_z^2}\right), \tag{4.3}$$

IV.2. Diseño del Simulador en Paralelo

Para el diseño de un programa paralelo no existe una receta o procedimiento establecido, sin embargo, hay varias propuestas que dan una idea lo suficientemente clara para la creación de estos programas. En esta sección se muestra el diseño del simulador paralelo, y para ello, se presenta el proceso de simulación que se debe realizar dentro del programa, posteriormente se mencionan las características y consideraciones que tendrá el algoritmo de simulación y, se enseña el diseño del programa desde la perspectiva presentada por **Foster (1995)**, así como el algoritmo en paralelo. Posterior al diseño se comentan algunos aspectos esenciales sobre descomposición de dominio implementados en el desarrollo del presente trabajo.

IV.2.1. Proceso de Simulación

Dentro del proceso de simulación podemos detectar áreas de cómputo paralelo y otras que sólo pueden ejecutarse de forma serial, por esto, a continuación se presentan de forma general las tareas que se realizan durante la simulación numérica de yacimientos:

i. Lectura de información. En esta fase se realiza la lectura de la información requerida para simular el comportamiento del yacimiento. Esta información incluye propiedades PVT de los tres componentes (aceite, gas y agua); tablas de permeabilidades relativas y presiones capilares de sistemas agua-aceite y gas-aceite; datos de permeabilidad absoluta y porosidad de la malla; propiedades del yacimiento como profundidad, temperatura, contactos de fluidos, datos de referencia, longitud, etc.; parámetros de simulación como tamaño de la malla, tiempo de simulación, geometría, formulación, etc.; y controles de simulación tanto de tiempo como numéricos para la solución del sistema de ecuaciones.

ii. Conversión de unidades. Consiste en homogeneizar las unidades que posee la información de entrada del simulador, de tal manera, que todos los términos que se calculen durante la ejecución del simulador (a partir de los datos) sean compatibles y ofrezcan buenos resultados dentro de un sistema de unidades especificado.

iii. Asignación de memoria. Esta fase se requiere para declarar y dimensionar todos los arreglos que se utilizan para calcular el comportamiento del fluido, la mayoría de los arreglos se dimensionan de acuerdo al tamaño de la malla y considerando si el sistema está fracturado o no, además, se dimensionan los arreglos requeridos para almacenar y resolver el sistema de ecuaciones, así como aquellos que contendrán la información relacionada a los pozos.

iv. Generación de la malla. Aquí se construye la malla de simulación sobre el yacimiento, esto es, se divide el volumen de roca en un conjunto de bloques cuya geometría dependerá del tipo de malla que se utilice, estos bloques se caracterizarán por tener propiedades de roca y fluidos propias.

v. Inicialización de la malla. En esta etapa se inicializan las variables primarias del simulador, para esto, se asignan los valores de p_o, s_g y s_w que se tienen al comienzo de la simulación por medio de un equilibrio capilar y gravitacional de fluidos, se utilizan las curvas de presión capilar para determinar la distribución vertical de saturaciones y las densidades de los fluidos para determinar la presión de las celdas partiendo de un punto de referencia y calculando la columna de fluido.

vi. Cálculo de las propiedades de la malla. Este punto implica el cálculo de las propiedades de la roca y fluidos para cada una de las celdas de la malla por medio de interpolaciones sobre los datos de entrada, entre estos datos encontramos los factores de volumen, densidades, viscosidades, presiones capilares, permeabilidades relativas, entre otras.

vii. Cálculo de los coeficientes de pozos. Esta tarea consiste en calcular todos los términos relacionados con el término fuente, desde la función de residuos hasta las derivadas con respecto a cada una de las variables, también, involucra el llenado parcial de la matriz de derivadas (jacobiano) y del vector independiente con sus correspondientes términos de acuerdo al número de celda que contiene el pozo.

viii. Cálculo de los coeficientes de las celdas. Esta fase se encarga del cálculo de los coeficientes de la matriz jacobiana y de los elementos del vector independiente a partir de las transmisibilidades y potenciales calculados con las propiedades de la malla interpoladas previamente, además, realiza las reducciones matriciales empleadas para el caso de simulaciones de yacimientos con el modelo de doble porosidad.

ix. Solución del sistema de ecuaciones. En esta etapa se resuelve el sistema de ecuaciones generado previamente y se obtiene el vector solución que está compuesto por incrementos de cada una de las variables (δp_o, δs_g, δs_w y δp_{wf} o δq_p).

x. Convergencia de la solución. Dado que la solución del sistema de ecuaciones es iterativa, esta parte del proceso consiste en verificar que la solución obtenida sea la correcta, esto se logra comparándola con una tolerancia establecida para cada una de las variables al inicio de la simulación.

Dentro del proceso de simulación descrito, de la etapa (vi) a (x) se entra en un ciclo de solución iterativa y en un ciclo de tiempo. El primero se usa para resolver el sistema de ecuaciones de forma iterativa, esto es, conforme se resuelve el sistema de ecuaciones (a un tiempo dado) se actualizan las variables y se vuelve a resolver el sistema hasta cumplir con cierta tolerancia. Una vez que se alcanza la tolerancia establecida, se tiene la solución del sistema de ecuaciones a ese tiempo, por lo que el paso siguiente es incrementar el tiempo y realizar los cálculos iterativos nuevamente.

IV.2.2. Estrategias de Paralelización

Una vez establecido el proceso de simulación, se requiere de su análisis para identificar las secciones que puedan ejecutarse en paralelo y determinar las estrategias de programación que deben de seguirse para alcanzar una aplicación paralela portable, que trabaje en sistemas de memoria distribuida, permita realizar simulaciones de gran escala y que sea escalable a múltiples procesadores.

De manera general, se presentó el proceso de simulación donde observamos que las operaciones a realizar, con excepción de las relacionadas con los pozos, se ejecutan sobre cada una de las celdas de forma indistinta, esto es, se sigue una metodología de cálculo sobre diferentes conjuntos de datos, esto sugiere que podemos realizar una descomposición del dominio físico, de tal forma que el problema original puede ser divido en un conjunto de problemas de menor tamaño, donde cada uno de ellos puede ser resuelto por un procesador, por lo cual, debemos dividir los datos entre los procesadores y a partir de ellos crear subproblemas cuya solución combinamos para obtener el resultado del problema original. Esto indica un alto grado de paralelismo, sin embargo, los cálculos relacionados a los pozos dificultan un poco el proceso de paralelización ya que se realizan únicamente sobre las celdas que contienen disparos, siendo conveniente evaluar la factibilidad de establecer algoritmos de balance de carga que consideren la repartición de las operaciones de la malla de simulación entre los procesadores, así como la localización de los pozos dentro de ella con la finalidad de evitar la sobrecarga de cálculos o grandes tiempos muertos, elementos que volverían al ineficiente programa.

Dentro del proceso de simulación observamos áreas con potencial de paralelización como: la entrada de datos, la inicialización del sistema, la creación de la malla, la construcción del jacobiano y las rutinas para resolver el sistema de ecuaciones. Una vez divididas estás etapas entre el número de procesadores, cada una requerirá de sus propios

recursos tanto de memoria como de procesamiento. Además, cada uno de los procesadores en paralelo, requerirá intercambiar información con aquellos que se ejecuten en dominios aledaños para obtener la solución de todo el problema.

IV.2.2.1. Modelo de Programación

Existen varias alternativas para organizar los programas paralelos, por lo que el modelo de programación empleado para el desarrollo del simulador en paralelo es híbrido, ya que para resolver este problema se requiere la combinación de elementos de varios modelos como el de datos paralelos y el de paso de mensajes, que se combinan para obtener un modelo SPMD en un sistema de memoria distribuida.

El uso de librerías de paso de mensajes, como MPI, permite escribir programas paralelos eficientes para sistemas de memoria distribuida. Estas librerías contienen rutinas para iniciar y configurar el ambiente de mensajes para enviar y recibir paquetes de datos, además, estas librerías permiten al usuario escribir programas que puede ejecutar en diferentes computadoras sin la necesidad de re-escribir el código, por lo que los objetivos de portabilidad, arquitectura y transparencia de red han sido alcanzados por esta implementación. Para escribir el simulador en paralelo usando paso de mensajes, se tienen que desarrollar cantidades considerables de código para manejar algunas tareas como la comunicación y sincronización entre procesadores, la partición y distribución de datos, mapeo de procesos a los procesadores y estructuras de datos de entrada y salida, todas ellas con la finalidad de transferir información parcial o final de procesamiento durante la ejecución del simulador.

En un sistema multiprocesador que ejecuta un conjunto de instrucciones, el paralelismo de datos se alcanzará cuando cada procesador realiza las mismas tareas sobre diferentes partes de los datos de simulación. El esquema de datos paralelos implica el dividir los datos del simulador entre los procesadores, para esto, se fragmenta la malla de cálculo del problema original en varias mallas de menor dificultad que conservan intacta la información de cada una de las celdas que contienen. Cada uno de los problemas que se obtienen posee su propia solución que se combina con las demás para alcanzar una solución única perteneciente al problema original. De acuerdo a lo anterior, la combinación de los esquemas mencionados nos permite crear una aplicación en paralelo robusta y eficiente.

Finalmente, para el desarrollo del simulador se trabaja con el esquema de hebras para observar el comportamiento del algoritmo en un procesador, este modelo consiste en la ejecución de tareas de forma concurrente y cooperando entre sí. Básicamente, la diferencia entre el esquema de hebras y el anterior está en el número de procesadores empleados durante la ejecución del simulador.

IV.2.2.2. Paradigma de Programación

El paradigma bajo el cual se diseña el simulador en paralelo es híbrido ya que se combinan elementos de paradigmas diferentes y, adicionalmente, porque la línea que existe entre cada uno de estos es difusa como se menciona en el capítulo III; los paradigmas que se utilizan son: maestro-esclavo, dividir y conquistar, y SPMD.

En el algoritmo de simulación en paralelo podemos emplear el paradigma maestro-esclavo ya que en ciertos momentos de la ejecución, requerimos el recopilar información calculada en un procesador (el cual consideramos como maestro), esta información se recibe del resto de los procesadores (considerados esclavos) por medio de mensajes a través de una red de comunicación y nos permite determinar la convergencia del sistema de ecuaciones del problema original durante cada iteración realizada dentro de cada paso de tiempo. De acuerdo a esto, se requiere el sincronizar el envío de resultados de los procesadores esclavos al procesador maestro, para que este pueda evaluar toda la información obtenida y determinar si se requiere una nueva iteración o pasar al siguiente nivel de tiempo. Dentro de este paradigma podemos limitar la escalabilidad del programa al incrementar el número de procesadores ya que generaríamos una sobrecarga de envío de información al nodo maestro, por lo que necesitamos implementar algoritmos que permitan reducir las comunicaciones en momentos clave.

Para el desarrollo del simulador paralelo también empleamos el paradigma de dividir y conquistar, éste es aplicable gracias a la naturaleza del problema ya que encontramos que podemos dividir al yacimiento por regiones que representen instancias del problema original pero cuya solución representa un menor grado de dificultad donde cada una de ellas puede obtenerse en cada uno de los procesadores disponibles. Este paradigma consiste en dividir el problema original en otros de tamaño menor, calcular la solución de cada uno de éstos y juntarlas para obtener un resultado final. Este esquema se parece al de maestro-esclavo con la diferencia en que no considera la presencia de un nodo maestro, sin embargo pueden combinarse para la construcción del simulador.

El empleo de los paradigmas anteriores genera una aplicación SPMD cuyas características son muy similares y permite la creación de programas eficientes siempre y cuando la distribución de los procesos y datos sea homogénea.

IV.2.3. Partición, Comunicación, Aglomeración y Mapeo

De acuerdo a la metodología presentada por **Foster (1995)**, se realizó la ***partición*** del problema en un conjunto de tareas que observamos previamente en el proceso de simulación. Dado que la estrategia de paralelización se basa en la descomposición de dominio por ser la más conveniente para la naturaleza del problema, las tareas se ejecutarán de forma simultánea en todos los procesadores sobre diferentes conjuntos de datos para realizar los cálculos respectivos a la malla.

Por otro lado, durante la ejecución del simulador se requiere de ***comunicación*** entre procesadores principalmente para transferir información durante la lectura de datos y la solución del sistema de ecuaciones, esta comunicación se realiza por medio del paso de mensajes a través de las librerías de MPI. Un aspecto primordial en la comunicación es la entrada y/o salida de datos ya que puede afectar el comportamiento del programa, los mensajes largos se emplean para mejorar la eficiencia de transferencia de redes y los comandos de entrada y/o salida especifican la comunicación entre dos procesos concurrentes. La comunicación ocurre entre dos procesos cuando un comando de entrada en uno de ellos hace referencia al otro como fuente, también cuando un comando de salida en otro proceso tiene como destino a otro proceso y cuando el objetivo del comando de

entrada de un proceso coincide con el de salida de otro proceso. Para la comunicación entre procesos se requiere sincronización de los comandos de entrada y salida, lo que significa que en la implementación tendrá que retrasarse cualquiera de los dos procesos que termine primero. La reducción del tiempo de comunicación se obtiene cuando se usa tecnología de comunicación asíncrona.

Después de la partición del problema y de establecer la comunicación dentro del diseño continuamos con la ***aglomeración*** de tareas para aumentar la granularidad de los procesos y reducir la comunicación entre ellos, sin embargo, en el diseño no existe una repartición específica de instrucciones sino de datos, por lo que la aglomeración se orienta más a la forma en que se repartirán los datos entre los procesadores para mantener un equilibrio de operaciones entre los procesadores.

Finalmente, el ***mapeo*** de tareas debe minimizar los costos de comunicación y en algoritmos de descomposición de dominio se realiza de forma relativamente sencilla y transparente, ya que en cada uno de los procesadores se ejecuta un mismo número de tareas por lo que la repartición de ellas depende únicamente de la topología de la red de comunicaciones de la computadora paralela.

IV.2.4. Algoritmo Paralelo

De forma general, el algoritmo del programa paralelo considerando la descomposición de dominio se presenta a continuación:

a. Lectura de datos de simulación por nodo maestro.

b. Partición de los datos de entrada en función del número de procesadores a emplear.

c. Distribución de los datos entre los procesadores.

d. Conversión de unidades por procesador.

e. Alojamiento de la memoria.

f. Generación de la malla de simulación para el dominio correspondiente.

g. Inicialización de las variables primarias.

h. Inicio del ciclo de tiempo e iterativo.

i. Cálculo de las propiedades petrofísicas y de fluidos de la malla.

j. Cálculo de las derivadas de los pozos.

k. Intercambio de información entre procesos, correspondiente al traslape de dominios.

l. Asignación de los coeficientes de los pozos en el jacobiano.

m. Cálculo de los coeficientes de la malla para el jacobiano.

n. Construcción del sistema de ecuaciones.

o. Solución del sistema de ecuaciones.

p. Verificación de la convergencia.

q. Recolección de resultados de los procesadores.

r. Distribución de resultados de convergencia.

s. Actualización de las variables primarias.

t. Incremento en el tiempo de simulación.

En el algoritmo presentado, la ejecución en paralelo se realiza a partir de la actividad "*c*" cuando cada uno de los procesadores disponen de información para trabajar, a partir de aquí cada procesador realiza las tareas subsecuentes de manera independiente hasta la solución del sistema de ecuaciones, después, se requiere enviar los resultados a un nodo maestro para verificar la convergencia y determinar si se requiere una nueva iteración o si se puede avanzar en el tiempo. Posteriormente, cada procesador continúa con su trabajo dependiendo de la información recibida.

IV.3. Descomposición de Dominio

La idea básica de descomposición de dominio consiste en que en lugar de resolver un gran problema en un dominio, puede ser más conveniente resolver varios problemas de menor tamaño en subdominios un cierto número de veces. El término descomposición de dominio tiene diferentes significados para especialistas dentro de la disciplina de PDE's los cuales pueden presentarse en una sola aplicación:

- *En cómputo en paralelo*, en ocasiones se refiere al proceso de distribuir datos de un modelo computacional entre los procesadores en una computadora de memoria distribuida, en este contexto se refiere a las técnicas de descomposición de estructuras de datos y puede ser independiente de los métodos de solución numérica.

- *En un análisis asintótico*, significa la separación del dominio físico en regiones que pueden ser modeladas con diferentes ecuaciones, con las interfaces entre los dominios manejadas por varias condiciones (ej. continuidad). En este contexto la descomposición de dominios se refiere a la determinación de cual PDE resolver.

- *En métodos de precondicionamiento*, se refiere al proceso de subdividir la solución de un gran sistema de ecuaciones en problemas más pequeños cuya solución puede

ser usada para generar un precondicionador para el sistema de ecuaciones que resulta de la discretización de la PDE en todo el dominio, en este contexto la descomposición de dominios se refiere sólo al método de solución del sistema algebraico de ecuaciones obtenido de la discretización.

Uno de los primeros métodos conocidos de descomposición de dominio fue introducido por Schwarz en 1870 y no fue concebido originalmente como un método numérico. El método alternado de Schwarz puede ser usado para resolver problemas de valor en la frontera en dominios que representan la unión de dos subdominios, resolviendo de forma alternada el mismo problema de valor en la frontera en cada uno de los dominios individuales. Este método es empleado para el desarrollo del presente trabajo con algunas modificaciones.

IV.3.1. Método Alternado de Schwarz

Para el método alternado de Schwarz consideramos un dominio como el mostrado en la figura IV.3, **Smith, Bjørstad y Gropp (1996)**, con $\Omega = \Omega_1 \cup \Omega_2$ donde buscamos resolver la ecuación en derivadas parciales.

$$\begin{aligned} Lu &= f \quad en\ \Omega \\ u &= g \quad en\ \partial\Omega\ , \end{aligned} \tag{4.4}$$

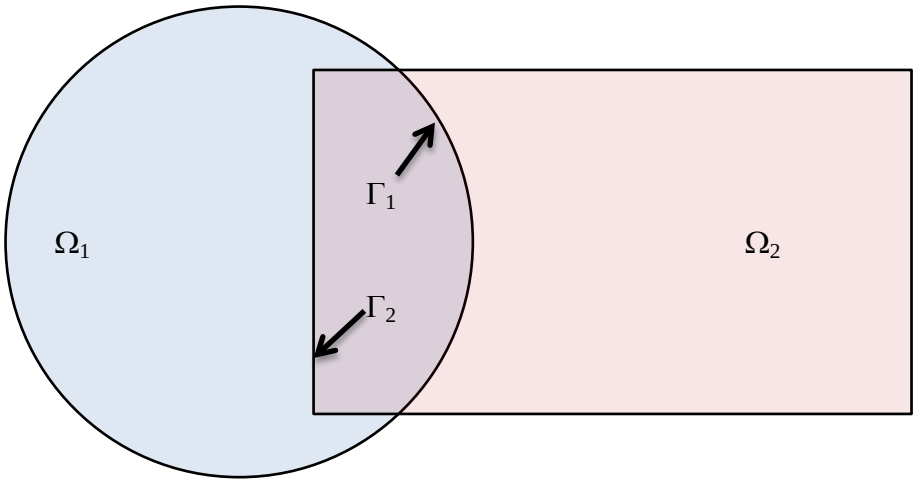


Fig. IV.3 Configuración de dominios para el método de Schwarz alternado

La forma simple de ***L*** es el menos Laplaciano, $-\Delta$, y por simplicidad restringimos la solución del problema a condiciones de frontera tipo Dirichlet, esto debido a que pueden emplearse condiciones de frontera más generales tipo Neumann. En la **Fig. IV.3**, $\partial\Omega$ representa la frontera de Ω, y se debe tener en cuenta que Ω_1 y Ω_2 no incluyen sus fronteras. Por otro lado, $\overline{\Omega} = \Omega \cup \partial\Omega$ representa todo el dominio. Las fronteras artificiales, Γ_i, son la parte de la frontera de Ω_i que se encuentra dentro de Ω. El resto de las fronteras

de los subdominios se denotan como $\partial\Omega_i \setminus \Gamma_i$; en otras palabras, son todos los puntos de $\partial\Omega_i$ que no están en Γ_i.

Para describir el método alternado de Schwarz se introduce la siguiente nomenclatura, consideremos que u_i^n representa la solución aproximada en $\overline{\Omega}_i$ después de *n* iteraciones y, $u_1^n|\Gamma_2$ es la restricción de u_1^n a Γ_2; de manera similar $u_2^n|\Gamma_1$ sea la restricción de u_2^n a Γ_1.

El método alternado de Schwarz inicia con la selección de una aproximación inicial, u_2^0, para los valores en Ω_2 (de hecho sólo se requieren valores a lo largo de Γ_1). Posteriormente, de forma iterativa para *n* = 1, 2, 3, … , se resuelve el problema de valor en la frontera:

$$\begin{aligned} Lu_1^n &= f && en\ \Omega_1 \\ u_1^n &= g && en\ \partial\Omega_1 \setminus \Gamma_1\ , \\ u_1^n &= u_2^{n-1} \mid \Gamma_1 && en\ \Gamma_1 \end{aligned} \tag{4.5}$$

para u_1^n. A continuación se resuelve el problema de valor en la frontera:

$$\begin{aligned} Lu_2^n &= f && en\ \Omega_2 \\ u_2^n &= g && en\ \partial\Omega_2 \setminus \Gamma_2\ , \\ u_2^n &= u_1^n \mid \Gamma_2 && en\ \Gamma_2 \end{aligned} \tag{4.6}$$

Por lo tanto, en cada paso intermedio del método alternado de Schwarz resolvemos el problema de valor en la frontera en el subdominio Ω_i con los valores de frontera dados, ***g***, en la frontera verdadera, $\partial\Omega_i \setminus \Gamma_i$, y la solución previa aproximada en el interior de la frontera Γ_i.

En cómputo numérico, se puede trabajar con alguna discretización dimensional de la PDE a resolver, esto nos puede llevar a dos tipos de problemas en los que la discretización de los dos dominios puede o no coincidir en sus nodos. Para nuestro caso consideramos mallas que coinciden, sin embargo, de no ser el caso se tendría que interpolar entre las soluciones de los dominios para obtener aproximaciones a los resultados correspondientes a cada tipo de malla.

IV.3.2. Método de Schwarz discretizado en mallas coincidentes

Consideremos el caso donde la discretización de Ω coincide en la zona de traslape y asumamos que los sistemas (4.5) y (4.6) se discretizaron usando diferencias finitas, **Smith, Bjørstad y Gropp (1996)**.

Entonces, u_i, es un vector con coeficientes:

$$u_i = \begin{pmatrix} u_{\Omega_i} \\ u_{\partial\Omega_i \setminus \Gamma_i} \\ u_{\Gamma_i} \end{pmatrix}, \tag{4.7}$$

donde los coeficientes $u_{\partial\Omega_i \setminus \Gamma_i}$ se conocen dado que se tienen condiciones de frontera tipo Dirichlet, estos se mantienen como "incógnitas" para simplificar la presentación. Asociado con el lado derecho, $\boldsymbol{f}$, y los valores de frontera, $\boldsymbol{g}$,sus equivalente discretizados son $\boldsymbol{f_i}$ y $\boldsymbol{g_i}$. La matriz $\boldsymbol{A_i}$ es la forma discreta del operador $\boldsymbol{L}$ restringido a Ω_i, y posee tres componentes, la matriz $\boldsymbol{A_{\Omega i}}$ que representa el arreglo de los nodos internos, $\boldsymbol{A_{\partial\Omega i \setminus \Gamma i}}$ que representa el acoplamiento entre los nodos internos y la frontera verdadera, $\partial\Omega_i \setminus \Gamma_i$ y $\boldsymbol{A_{\Gamma i}}$ que representa el acoplamiento entre los nodos internos y los nodos que yacen en la frontera artificial Γi. La matriz $\boldsymbol{A_i}$ es rectangular con un renglón por cada nodo interior y una columna por cada nodo (incluyendo los de la frontera). Esto puede ser escrito como $A_i = (A_{\Omega i} \;\; A_{\partial\Omega i \setminus \Gamma i} \;\; A_{\Gamma i})$.

Los dos subproblemas en el método alternado de Schwarz pueden ser escritos en forma discreta como:

$$\begin{aligned} &A_1 u_1^n = f_1 && en\ \Omega_1 \\ &u^n_{\partial\Omega_1 \setminus \Gamma_1} = g_1 && en\ \partial\Omega_1 \setminus \Gamma_1 \\ &u^n_{\Gamma_1} = u^{n-1}_{\Omega_2} \mid \Gamma_1 && en\ \Gamma_1 \end{aligned}, \tag{4.8}$$

para u_1^n. A continuación se resuelve el problema de valor en la frontera:

$$\begin{aligned} &A_2 u_2^n = f_2 && en\ \Omega_2 \\ &u^n_{\partial\Omega_2 \setminus \Gamma_2} = g_2 && en\ \partial\Omega_2 \setminus \Gamma_2 \\ &u^n_{\Gamma_2} = u^n_{\Omega_1} \mid \Gamma_2 && en\ \Gamma_2 \end{aligned}, \tag{4.9}$$

A partir de estas ecuaciones podemos establecer la versión discreta del método alternado de Schwarz:

Algoritmo del método alternado de Schwarz

- $w_1^0 \leftarrow 0$
- Para n=1 …..
    - Resolver para u_1^n

$$A_1 u_1^n = f_1 \qquad en\ \Omega_1$$
$$u^n_{\partial\Omega_1 \setminus \Gamma_1} = g_1 \qquad en\ \partial\Omega_1 \setminus \Gamma_1$$
$$u^n_{\Gamma_1} = w_1^{n-1} \qquad en\ \Gamma_1$$

- $w_2^n \leftarrow u^n_{\Omega_1} | \Gamma_2$
- Resolver para u_2^n

$$A_2 u_2^n = f_2 \qquad en\ \Omega_2$$
$$u^n_{\partial\Omega_2 \setminus \Gamma_2} = g_2 \qquad en\ \partial\Omega_2 \setminus \Gamma_2$$
$$u^n_{\Gamma_2} = w_2^n \qquad en\ \Gamma_2$$

- $w_1^n \leftarrow u^n_{\Omega_2} | \Gamma_1$
- $Si\ \left||w_1^n - w_1^{n-1}|\right| \leq tol_{\Gamma_1}\ \ y\ \left||w_2^n - w_2^{n-1}|\right| tol_{\Gamma_2}\ \ parar$
- $Si\ \left||u_1^n - u_1^{n-1}|\right| \leq tol_{\Omega_1}\ \ y\ \left||u_2^n - u_2^{n-1}|\right| \leq tol_{\Omega_2}\ \ parar$

- Fin de ciclo

Al final del cómputo, $u^n_{\Omega_1}$ contiene la solución discreta aproximada para Ω_1 y $u^n_{\Omega_2}$ contiene la solución discreta aproximada para Ω_2. Para la zona de traslape uno es libre de usar cualquiera de las soluciones, o un promedio dado que ambas soluciones convergen al mismo valor.

Existen muchas maneras para determinar cuando el método alternado de Schwarz converge. En el algoritmo presentado se eligió que las soluciones aproximadas en los dominios deben cambiar menos que una tolerancia establecida con respecto a la iteración previa.

Finalmente, en este caso es conveniente mencionar que en cada paso intermedio basta con actualizar el resultado en la frontera de los subdominios, Γ_i, y no en las celdas con traslape.

IV.3.3. Múltiples Dominios

Actualmente, las máquinas paralelas poseen de docenas a miles de procesadores independientes y claramente los algoritmos de descomposición de dominio que consideran dos subdominios no son capaces de tomar ventaja de ellas. Por fortuna, las ideas de descomposición de dominio empleando los métodos de Schwarz conducen inmediatamente a métodos que involucran más de dos subdominios.

El método alternado de Schwarz por sí sólo tiene un bajo potencial de paralelización el cual puede apreciarse en su algoritmo, sin embargo, al observar un dominio discretizado en múltiples subdominios apreciamos que algunos de ellos poseen puntos comunes en la malla, esto implica que su solución puede actualizarse de manera simultánea en paralelo, lo anterior puede visualizarse en la **Fig. IV.4** donde se muestra la discretización del dominio global en múltiples subdominios.

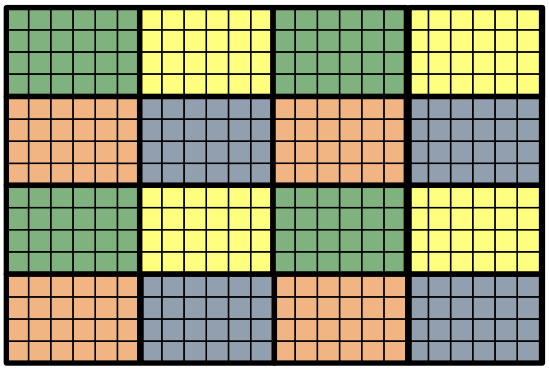


Fig. IV.4 Discretización en múltiples subdominios de un dominio global

En la **Fig. IV.4**, desde un punto de vista general, podemos definir una escala de colores de los subdominios donde a cada uno de ellos asociamos un color, de tal forma que dos subdominios que tengan puntos en común no tengan el mismo color, posteriormente, realizamos un ciclo de solución en función de los colores utilizados. El ritmo de convergencia dependerá del número de colores, entre menos se tengan, más rápido será la convergencia, sin embargo, dado que emplear un número de colores mínimo en los subdominios es difícil, uno puede emplear varios para asegurar el balance de carga de trabajo y datos entre los procesadores, esto idealmente deteriora el ritmo de convergencia ligeramente.

Si los subdominios son rectangulares, como en la Fig. IV.4, el número de colores mínimo requerido en dos dimensiones es de cuatro y en tres dimensiones de ocho. En este trabajo se emplea un ordenamiento de dominios conocido como “red-black” donde se consideran dos colores y la descomposición en subdominios se hace en una sola dirección, **Fig. IV.5**.

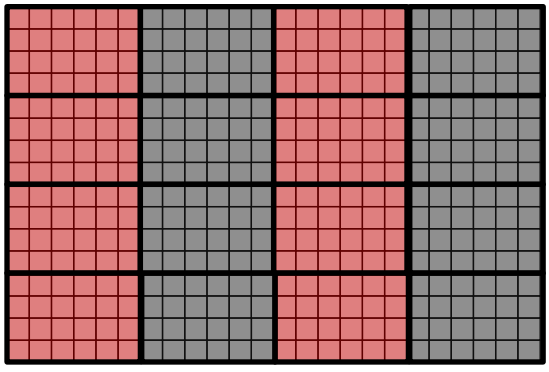


Fig. IV.5 Discretización en múltiples subdominios considerando dos colores

IV.3.4. Aspectos relevantes de métodos de descomposición de dominios

Durante la implementación de métodos de descomposición de dominios es importante tener en consideración los aspectos siguientes:

- Separar aquellos elementos que se refieran a cualquier parte del proceso y aquellos que se refieran en específico a la sección en paralelo, muchos elementos son ajenos a la paralelización, algunos otros como la solución de dominios por colores no lo es.

- Un aspecto importante para tener un uso eficiente de los procesadores en paralelo consiste en preservar la *localidad de datos.* Esto se debe a que en la mayoría de las computadoras en paralelo el tiempo que se toma en transferir datos es mucho mayor que el tiempo requerido para realizar cálculos numéricos, por ejemplo, el tiempo para enviar un valor real de doble precisión entre dos procesadores puede ser más de 50 veces de lo que tomaría multiplicar dos valores de doble precisión que se encuentren en un mismo procesador. Esta relación es en ocasiones empleada para el análisis el comportamiento de algoritmos paralelos, una relación alta requiere de un mayor trabajo local para alcanzar una buena eficiencia y, por lo tanto, en ocasiones implica que cada procesador necesita relativamente mayor memoria local.

- Dentro de un programa en paralelo, en lo que se refiere al mecanismo de solución de los subdominios se tiene que su característica esencial es la habilidad para actualizar el lado derecho y condiciones de frontera, y devolver la solución en diferencias a cualquier punto del dominio.

- Una de las partes más complejas de un programa con descomposición de dominio es aquella referida al almacenamiento de datos para cada dominio. La causa de esto es el hecho de que en dominios con traslape no se tiene una estructura simple de datos que pueda representar los datos de manera global para todo el dominio y los elementos de los subdominios individuales.

- Una de las propiedades clave para las EDP elípticas es que su solución en cualquier punto es influenciada por todos los valores de frontera sin importar su distancia, esto significa que para cualquier aproximación en diferencias, el método iterativo puede no converger a una solución correcta hasta que se tenga la oportunidad de que todos los datos de las fronteras afecten los nodos internos.

IV.3.5. Métricas de Programas Paralelos

La mejor forma de medir el progreso de un programa en paralelo es comparándolo con una versión serial del mismo, es decir, una versión hecha para ejecutarse empleando un solo procesador y que por lo tanto realizará todas las tareas requeridas, dicho de otra manera, es una versión en la que un cálculo puede realizarse solamente después de que otro ha finalizado. El objetivo del cómputo en paralelo debe ser el obtener resultados de forma rápida y económica de problemas de mayor tamaño que los obtenidos empleando métodos convencionales de cómputo, esto se logra cuando parte de los cálculos son independientes de los demás por lo que pueden hacerse de manera simultánea.

El objetivo de la programación en paralelo no es únicamente el optimizar la velocidad del programa, también se debe optimizar el tiempo de ejecución de la aplicación, los requerimientos de memoria, costos de implementación y mantenimiento, etc., y para tomar decisiones de desarrollo relacionadas a estos aspectos, partimos de la respuesta que obtenemos de modelos matemáticos que nos permiten medir la eficiencia de nuestro algoritmo. Con estos modelos podemos comparar la eficiencia de diferentes algoritmos, evaluar la escalabilidad e identificar cuellos de botella y otras ineficiencias.

Las métricas para evaluar el comportamiento pueden ser tan diversas como el tiempo de ejecución, eficiencia paralela, requerimientos de memoria, latencia, velocidades de entrada/salida, costos de diseño, costos de implementación, costos de verificación, potencial de reuso, requerimientos de hardware, manejo de la red, costos de hardware, costos de mantenimiento, portabilidad y escalabilidad. La importancia relativa de estas métricas variará de acuerdo a la naturaleza del problema, a continuación se describen algunas de las más importantes.

El tiempo de ejecución de un programa paralelo, se define como el tiempo transcurrido desde que el primer procesador comienza su ejecución sobre el problema hasta que el último procesador termina su trabajo. Sin embargo, durante la ejecución cada procesador realiza cálculos computacionales, comunicaciones, o puede encontrarse en espera, por lo tanto, el tiempo total de ejecución T puede definirse como la suma del cómputo, comunicaciones y tiempo de espera de un procesador, o como la suma de los tiempos mencionados de todos los procesadores, dividida por el número de procesadores.

La eficiencia y velocidad son otras métricas usadas para evaluar algoritmos paralelos, la primera es la fracción de tiempo que los procesadores emplean para realizar trabajo útil y caracteriza la efectividad con la que un algoritmo usa los recursos computacionales de una máquina paralela. La velocidad es un factor por el cual el tiempo de ejecución es reducido en P procesadores.

Un aspecto importante dentro del análisis de escalabilidad, es el estudio de cómo se comporta el algoritmo paralelo con parámetros como el tamaño del problema, conteo de procesos y costo de mensajes. En particular, podemos evaluar la escalabilidad de un algoritmo determinando su efectividad al incrementar el número de procesadores, para esto, podemos realizar el análisis de un problema fijo donde obtenemos el tiempo de ejecución y eficiencia en función del número de procesadores, con lo cual se puede determinar el mayor

número de procesadores que se pueden usar para resolver de forma eficiente el problema o el numero de procesadores requeridos para mantener una eficiencia específica. Además, en ocasiones es conveniente evaluar la escalabilidad del programa con el tamaño del problema escalado, donde, la cantidad de cómputo debe escalarse con el número de procesadores para mantener constante la eficiencia.

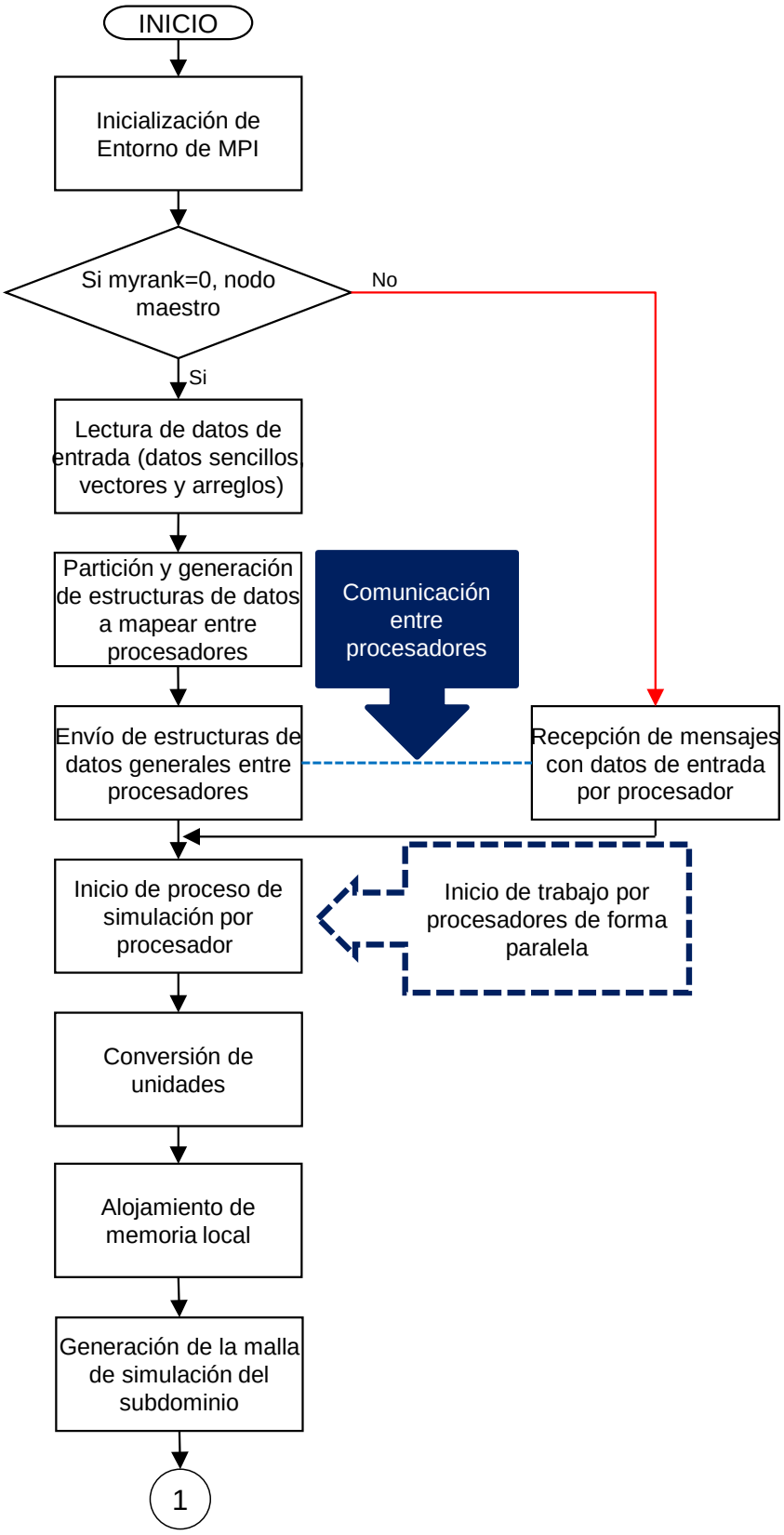
Un factor determinante en el comportamiento de muchos programas paralelos es el tiempo requerido para transmitir datos entre la memoria y un almacenamiento secundario, esto es, el tiempo requerido para entrada/salida. Dado que las peticiones de entrada/salida tienden a ser más caras que las comunicaciones interprocesador, en ocasiones es mejor realizar una redistribución explícita de datos en memoria para minimizar el número de comunicaciones.

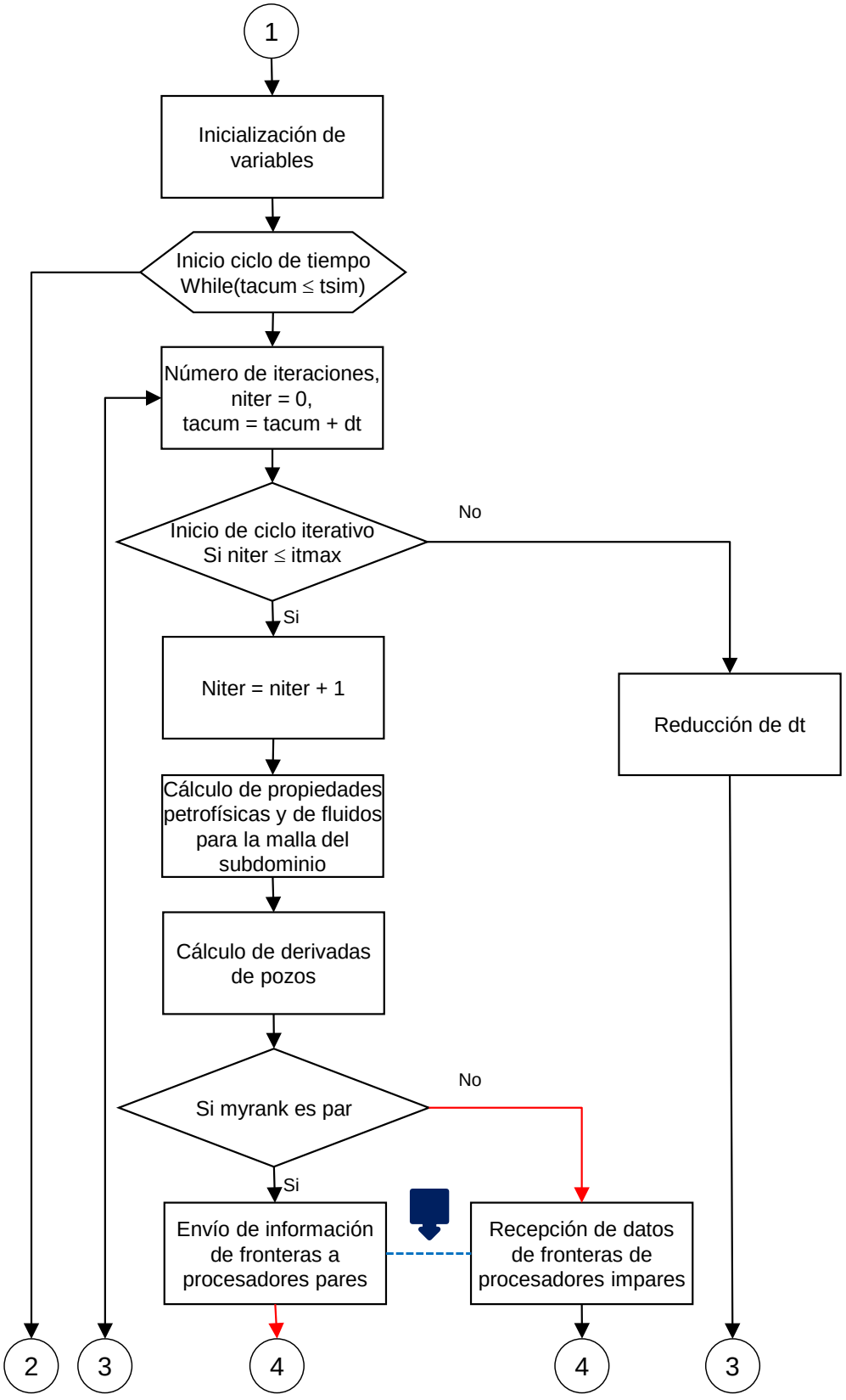
IV.4. Diagrama de flujo del simulador

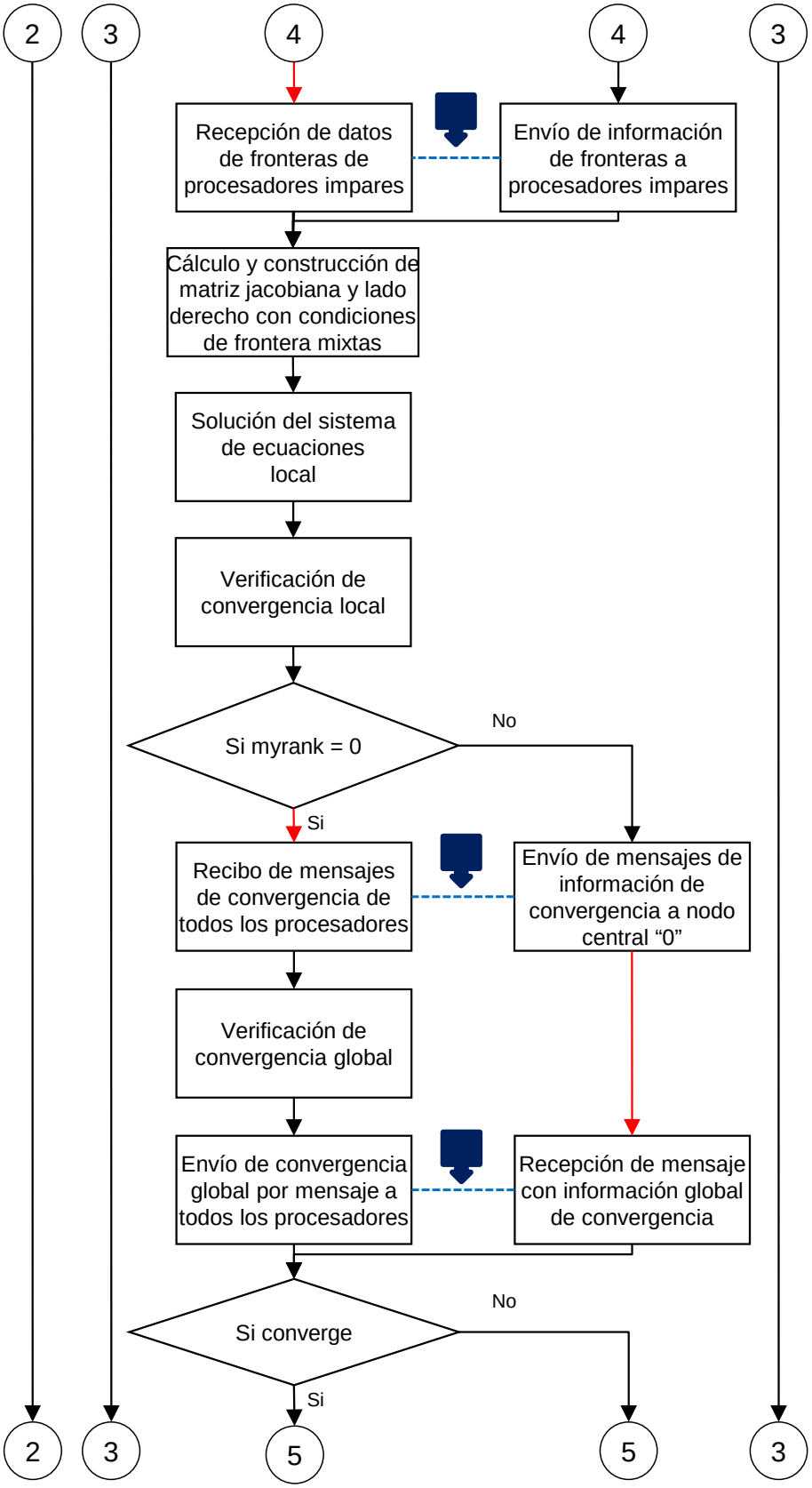
El diagrama de flujo del simulador en paralelo con descomposición de dominio se presenta a continuación en la **Fig. IV.6**, éste se deriva del algoritmo definido en la sección IV.2.4., en él se muestran las fases del proceso de simulación de manera general, desde la inicialización del entorno de MPI hasta el desalojo de la memoria local de cada uno de los procesadores, pasando por cada etapa previamente mencionada.

Un aspecto a destacar es la comunicación entre procesadores, la cual se muestra con líneas discontinuas en color azul y donde se tiene el envío y recepción de mensajes con paquetes de información durante la ejecución del proceso. Es conveniente mencionar que a partir del primer envío de mensajes, que corresponden a los datos de entrada, todos los procesadores inician la ejecución de sus respectivas tareas. El envío de datos se utiliza inicialmente para la distribución de la información de entrada al simulador a cada uno de los procesadores, posteriormente se emplea para el intercambio de información concerniente a las condiciones de frontera a presión constante requeridas por la descomposición de dominio, para finalmente ser usadas en la evaluación de la convergencia global de la solución del problema a partir de la convergencia local de cada subproblema.

Finalmente, dentro del diagrama de flujo se aprecian líneas continuas en color rojo, las cuales están asociadas al envío-recepción de mensajes, y que representan tiempos de ocio de aquellos procesadores que se encuentran esperando información para continuar con su ejecución.


INICIO
Inicialización de Entorno de MPI
Si myrank=0, nodo maestro
No
Si
Lectura de datos de entrada (datos sencillos vectores y arreglos)
Partición y generación de estructuras de datos a mapear entre procesadores
Comunicación entre procesadores
Envío de estructuras de datos generales entre procesadores
Recepción de mensajes con datos de entrada por procesador
Inicio de proceso de simulación por procesador
Inicio de trabajo por procesadores de forma paralela
Conversión de unidades
Alojamiento de memoria local
Generación de la malla de simulación del subdominio
1


1
Inicialización de variables
Inicio ciclo de tiempo
While(tacum ≤ tsim)
Número de iteraciones,
niter = 0,
tacum = tacum + dt
Inicio de ciclo iterativo
Si niter ≤ itmax
No
Si
Niter = niter + 1
Reducción de dt
Cálculo de propiedades petrofísicas y de fluidos para la malla del subdominio
Cálculo de derivadas de pozos
Si myrank es par
No
Si
Envío de información de fronteras a procesadores pares
Recepción de datos de fronteras de procesadores impares
2
3
4
4
3


2
3
4
4
3
Recepción de datos de fronteras de procesadores impares
Envío de información de fronteras a procesadores impares
Cálculo y construcción de matriz jacobiana y lado derecho con condiciones de frontera mixtas
Solución del sistema de ecuaciones local
Verificación de convergencia local
Si myrank = 0
No
Si
Recibo de mensajes de convergencia de todos los procesadores
Envío de mensajes de información de convergencia a nodo central “0”
Verificación de convergencia global
Envío de convergencia global por mensaje a todos los procesadores
Recepción de mensaje con información global de convergencia
Si converge
No
Si
2
3
5
5
3

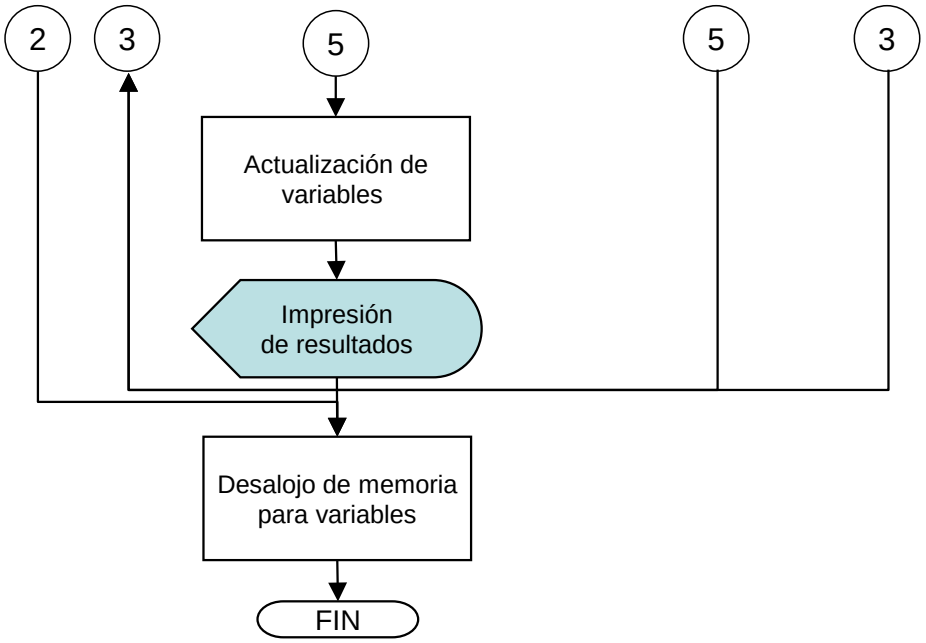


Fig. IV.6 Diagrama de flujo del simulador numérico de yacimientos en paralelo

V. CASOS DE APLICACIÓN

Como parte final de este trabajo se presenta a continuación un par de casos de aplicación, el primero de ellos para un yacimiento homogéneo y el segundo un caso de doble porosidad. Estos casos son sintéticos y a través de ellos se pretende evaluar el rendimiento del simulador en paralelo desarrollado bajo los lineamientos establecidos anteriormente.

V.1 Modelo de simulación de aceite negro en un medio homogéneo

Para este caso se creó un escenario de simulación de aceite negro en un yacimiento homogéneo con la finalidad de realizar sensibilidades en torno al número de procesadores y correlacionarlos con el tiempo de proceso para evaluar el rendimiento del algoritmo. Los principales datos de entrada se presentan en la **Tabla V.1**, la información PVT de los fluidos en la **Tabla V.2 (a, b y c)** y las curvas de permeabilidad relativa en la **Tabla V.3**.

Tabla V.1 Datos de entrada

Nx	500
Ny	100
Nz	1
Hx (m)	5,000
Hy (m)	1,000
Hz (m)	45
Tiempo de simulación	2,000
Número de pozos	2
Tipo de pozos	Productores
Gasto (bpd)	1,100
Temperatura de yacimiento (°C)	93
Presión de referencia (MPa)	33.095
Profundidad de referencia (m)	-2560.32
Porosidad (fr)	0.15
Permeabilidad X-Y (mD)	50
Permeabilidad Z (mD)	10
Sw irreductible	0.12

Tabla V.2a Información PVT aceite

P(MPa)	Bo(V/V)	μo (cp)	ρo (kg/m3)	Rs (m3/m3)
0.101353	1.062	1.040	740.74678	0.178095
1.825042	1.150	0.975	697.50471	16.117596
3.548731	1.207	0.910	677.37179	32.057098
6.996110	1.295	0.830	656.81460	66.073240
13.890867	1.435	0.695	624.62887	113.268411
17.338245	1.500	0.641	613.56586	138.023614
20.785624	1.565	0.594	605.18752	165.628337
27.680381	1.695	0.510	593.41538	226.180632
62.154166	1.579	0.740	637.01018	226.180632

Tabla V.2b Información PVT gas

P(MPa)	Bg(V/V)	μg (cp)	ρg (kg/m3)	Rv (m3/m3)
0.101353	0.93583	0.0080	1.03625	0.00000
1.825042	0.06790	0.0096	14.28162	0.00000
3.548731	0.03523	0.0112	27.52752	0.00000
6.996110	0.01795	0.0140	54.02178	0.00000
13.890867	0.00906	0.0189	107.00597	0.00000
17.338245	0.00727	0.0208	133.46804	0.00000
20.785624	0.00606	0.0228	159.91448	0.00000
27.680381	0.00455	0.0268	212.95640	0.00000
62.154166	0.00217	0.0470	447.42912	0.00000

Tabla V.2c Información PVT agua

P(MPa)	Bw(V/V)	μw (cp)	ρw (kg/m3)
0.101353	1.04188	0.31	996.10662
1.825042	1.04107	0.31	996.88158
3.548731	1.04027	0.31	997.65716
6.996110	1.03865	0.31	999.21014
13.890867	1.03542	0.31	1002.32343
17.338245	1.03381	0.31	1003.88373
20.785624	1.03221	0.31	1005.44648
27.680381	1.02900	0.31	1008.57929
62.154166	1.01312	0.31	1024.38975

Tabla V.3a Permeabilidad relativa gas aceite

Sg(-)	Krg(-)	Krog(-)
0.000	0.000	1.000
0.022	0.002	0.968
0.049	0.007	0.921
0.085	0.019	0.842
0.120	0.034	0.700
0.200	0.075	0.350
0.250	0.125	0.200
0.307	0.190	0.110
0.400	0.410	0.054
0.450	0.600	0.027
0.500	0.720	0.014
0.599	0.870	0.004
0.700	0.940	0.000
0.850	0.980	0.000
0.880	0.984	0.000
1.000	1.000	0.000

Tabla V.3b Permeabilidad relativa agua aceite

Sw(-)	Krw(-)	Krow(-)
0.160	0.000	1.000
0.200	0.002	0.650
0.300	0.021	0.388
0.400	0.056	0.220
0.500	0.113	0.100
0.600	0.209	0.045
0.700	0.324	0.017
0.800	0.439	0.006
0.840	0.503	0.000
1.000	1.000	0.000

En la gráfica de la **Fig. V.I** se presenta el comportamiento de la presión de fondo fluyendo para uno de los pozos y la relación gas aceite para la versión serial y en paralelo, en ellas se aprecia un buen ajuste con lo cual se valida el buen funcionamiento del algoritmo entre versiones.

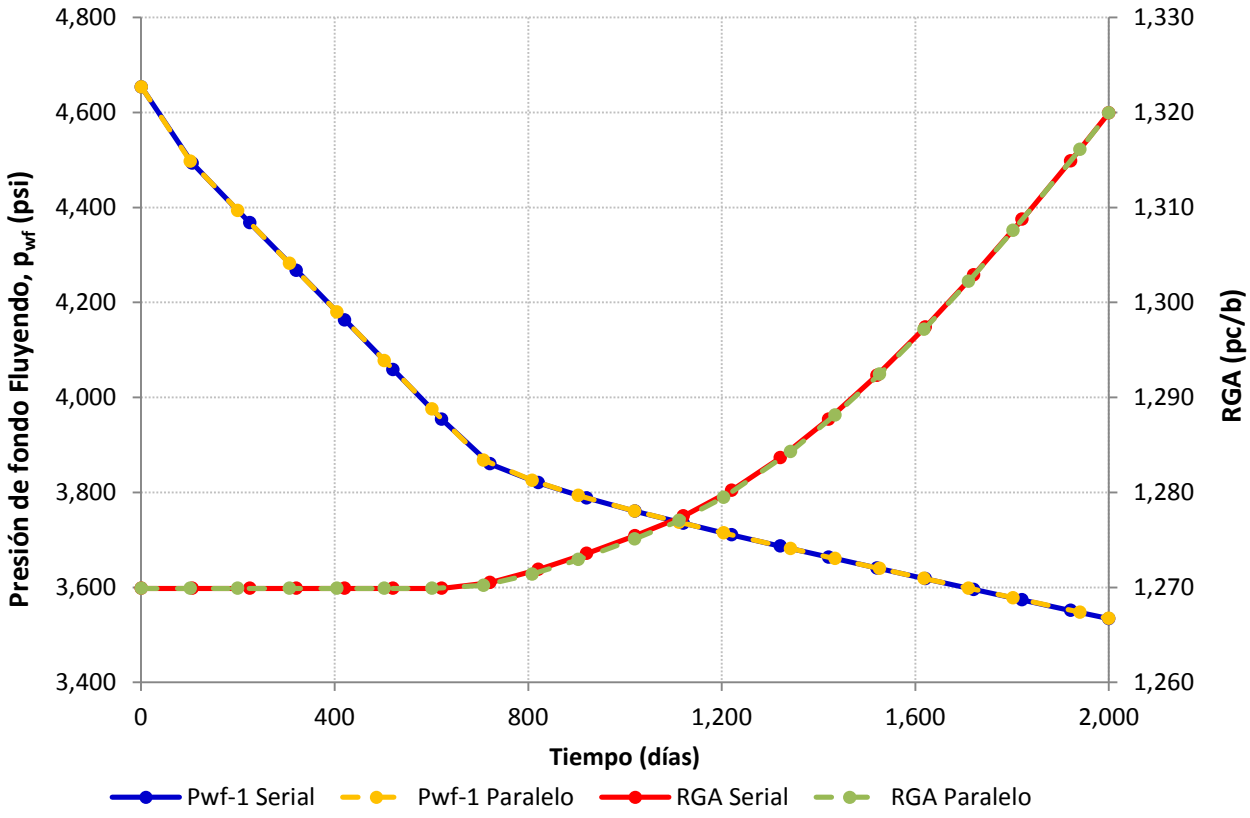


Figura V.1 Comportamiento de Pwf y RGA para el pozo-1 en la versión serial y en paralelo

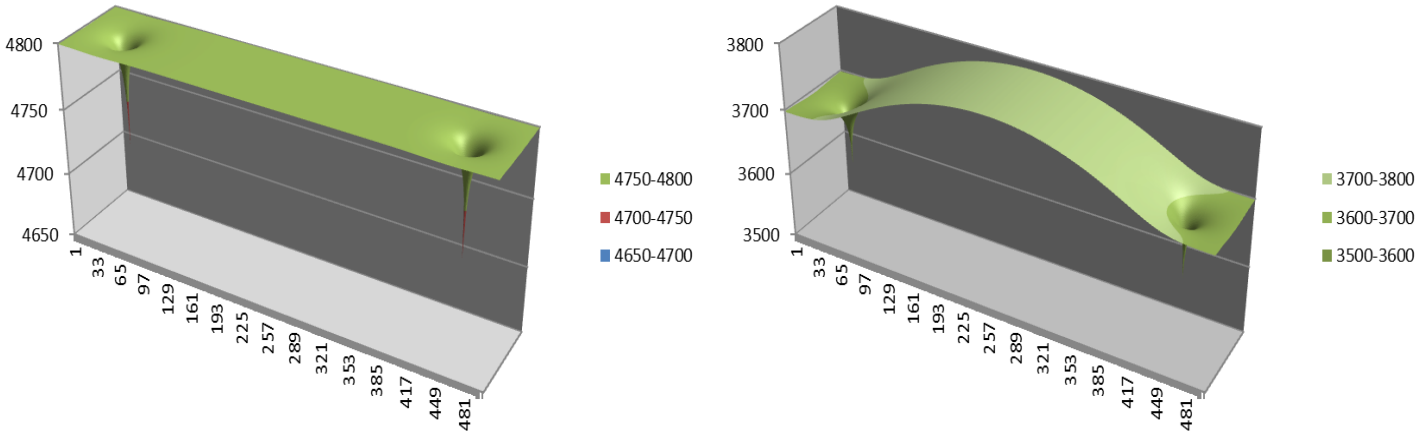


Figura V.2 Comportamiento de Presión en el modelo para t=1 día (izquierda), t=2000 días (derecha)

El caso de simulación contiene 50,000 y genera un sistema de ecuaciones de 150,000 variables que en una versión serial tarda en resolverse 405 segundos, para diferente número de procesadores los resultados en tiempo son los siguientes:

Tabla V.4 Resultados de procesamiento para diferente número de procesadores

Número de procesadores	*Tiempo (segundos)*	*Aceleración*	*Eficiencia*
2	246	1.6463	0.8232
4	189	2.1428	0.5357
8	178	2.2753	0.2844

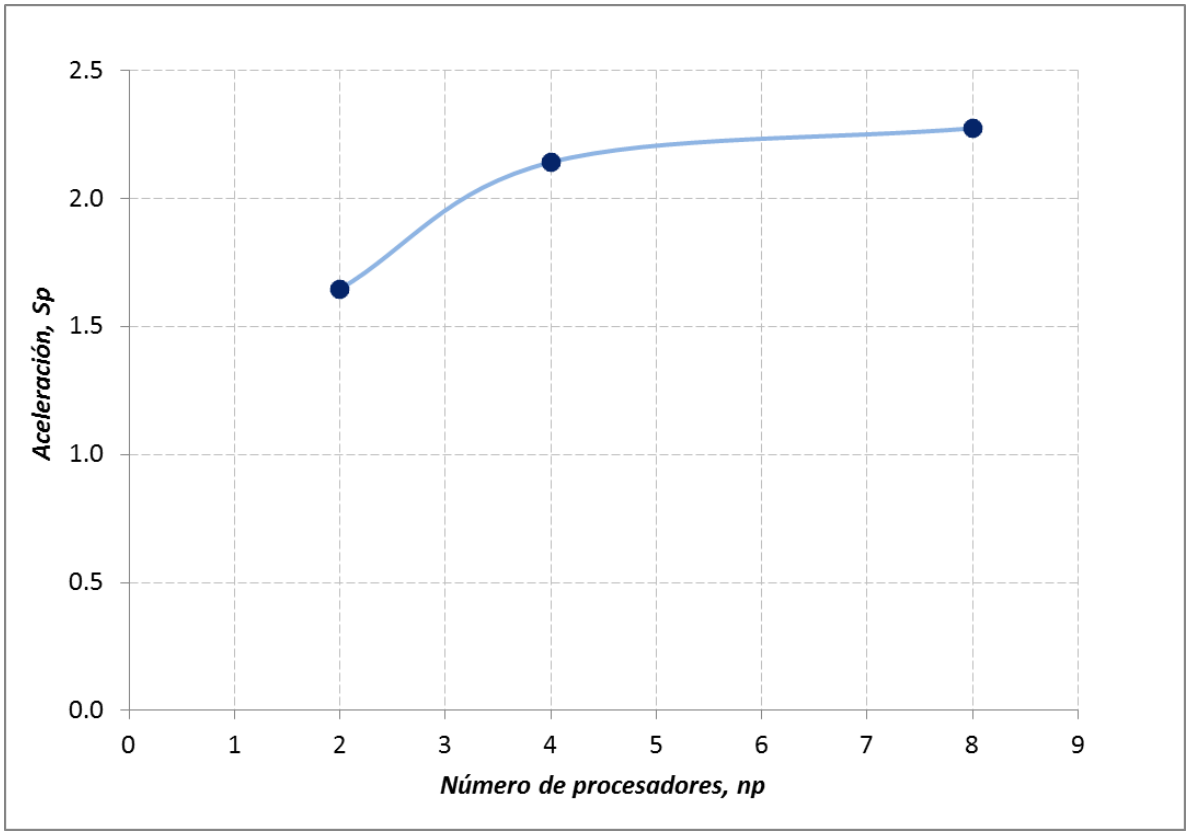


Figura V.3 Aceleración del simulador en paralelo

V.2 Modelo de simulación de aceite negro en un medio de doble porosidad

Este ejemplo se generó tomando como base la información del ejemplo anterior y se anexaron los datos correspondientes al segundo medio, la fractura, a continuación se presenta la información adicional

Tabla V.5 Datos de entrada	
Nx	1000
Ny	100
Nz	1
Hx (m)	5,000
Hy (m)	1,000
Hz (m)	45
Tiempo de simulación	400
Número de pozos	4
Tipo de pozos	Productores
Gasto (bpd)	1,740
Temperatura de yacimiento (°C)	93
Presión de referencia (MPa)	33.095
Profundidad de referencia (m)	-2560.32
Porosidad matriz (fr)	0.10
Porosidad fractura	0.05
Permeabilidad matriz X-Y (mD)	50
Permeabilidad matriz Z (mD)	10
Permeabilidad Fractura X-Y (mD)	200
Permeabilidad Fractura Z (mD)	100

Tabla V.6a Permeabilidad relativa gas aceite		
Sg(-)	**Krg(-)**	**Krog(-)**
	Matriz	
0.000	0.000	1.000
0.022	0.002	0.968
0.049	0.007	0.921
0.085	0.019	0.842
0.120	0.034	0.700
0.200	0.075	0.350
0.250	0.125	0.200
0.307	0.190	0.110
0.400	0.410	0.054
0.450	0.600	0.027
0.500	0.720	0.014
0.599	0.870	0.004
0.700	0.940	0.000
0.850	0.980	0.000
0.880	0.984	0.000
1.000	1.000	0.000
	Fractura	
0.000	0.000	1.000
1.000	1.000	0.000

Tabla V.6b Permeabilidad relativa agua aceite		
Sw(-)	Krw(-)	Krow(-)
	Matriz	
0.160	0.000	1.000
0.200	0.002	0.650
0.300	0.021	0.388
0.400	0.056	0.220
0.500	0.113	0.100
0.600	0.209	0.045
0.700	0.324	0.017
0.800	0.439	0.006
0.840	0.503	0.000
1.000	1.000	0.000
	Fractura	
0.000	0.000	1.000
1.000	1.000	0.000

Se presenta el resultado obtenido para la presión de fondo fluyendo de uno de los pozos productores así como su comparación con software comercial así como su visualización de la presión en todo el dominio. La versión serial demoró 1027 segundos en correr mientras la versión en paralelo 729 segundos.

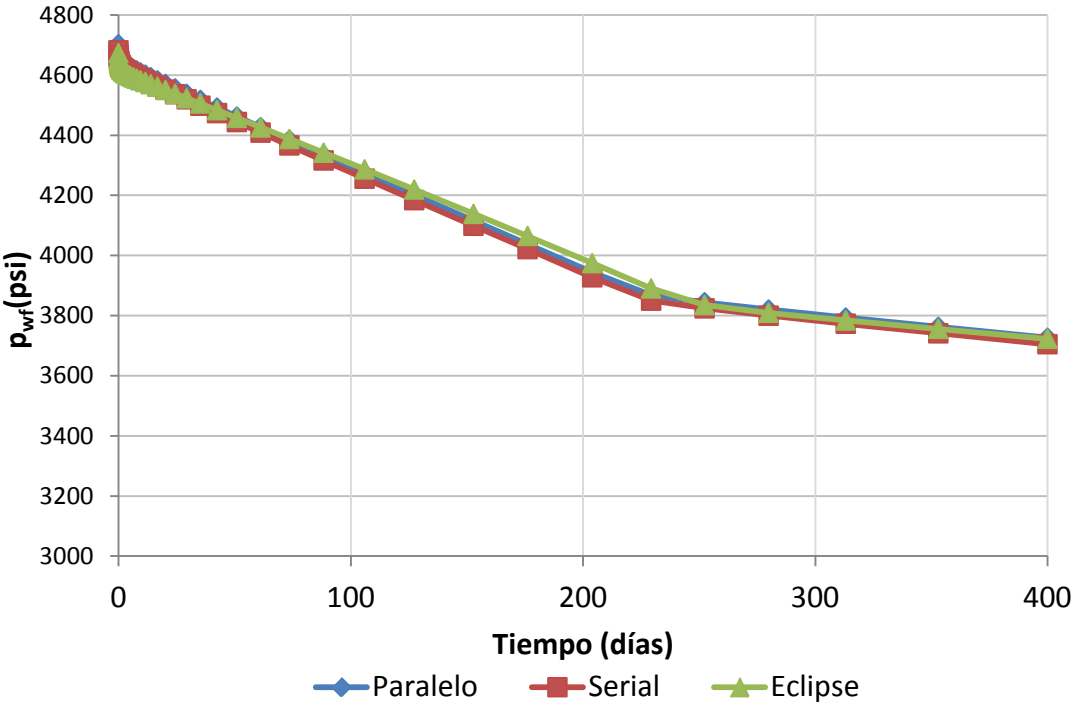


Figura V.4 Comportamiento de Presión de fondo fluyendo en el pozo 1 y comparación con la versión serial y Eclipse

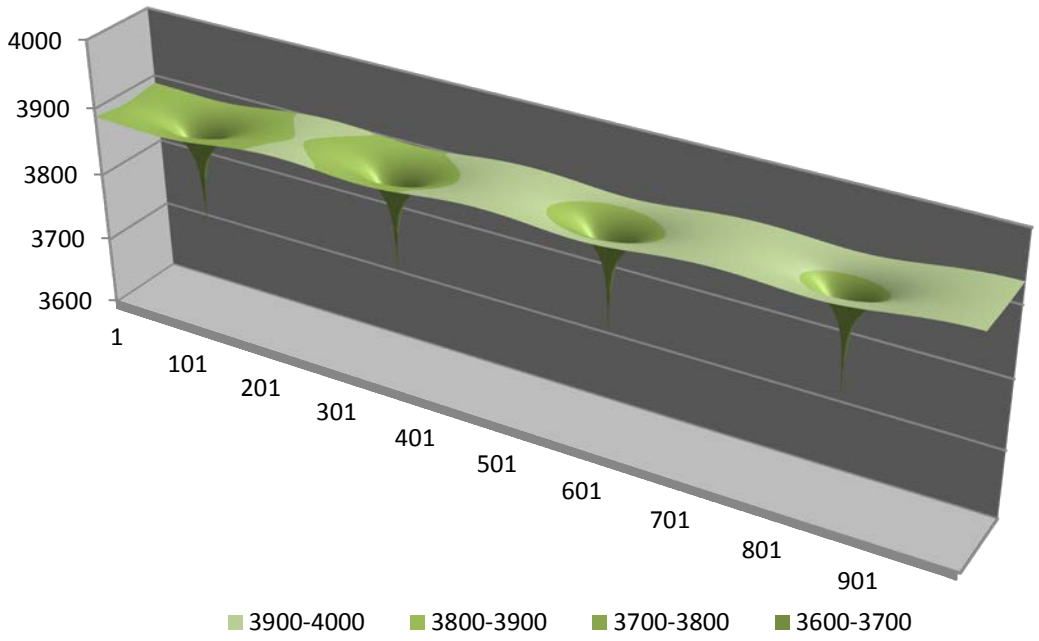


Figura V.5 Comportamiento de Presión en el modelo para t = 400 días

VI. CONCLUSIONES Y RECOMENDACIONES

En este trabajo se presenta el desarrollo de un simulador numérico de yacimientos en paralelo, empleando el modelo de aceite negro con el método de diferencias finitas, el cual es capaz de representar el comportamiento de yacimientos naturalmente fracturados en condiciones de flujo isotérmicas y su formulación es totalmente implícita, por lo que podemos concluir lo siguiente:

1. Se muestra el desarrollo numérico del simulador con una formulación totalmente implícita, para mantener su estabilidad tanto en las ecuaciones de flujo como en el modelo del pozo, con lo que se puede establecer que el error que se genera al realizar grandes corridas o emplear largos pasos de tiempo, es reducido.

2. Se presenta la combinación de la programación en paralelo, que pertenece al área de cómputo científico, con la ingeniería petrolera para obtener una herramienta poderosa empleada en el desarrollo y administración de campos.

3. Se desarrolló un simulador numérico en paralelo empleando librerías de paso de mensajes (MPI), que le permiten ser portable y con un alto grado de escalabilidad.

4. El simulador de yacimientos se elaboró empleando técnicas de descomposición de dominio y el mayor esfuerzo de programación se centró en la solución del sistema de ecuaciones, así como en la comunicación entre procesadores.

5. El simulador en paralelo se programó para su uso en máquinas paralelas, así como en computadoras con núcleos múltiples, por lo que puede ejecutarse en una mayor variedad de equipos de acceso fácil por los usuarios.

6. La solución de una EDP siempre se ve afectada por todas las fronteras.

7. El método de Schwarz alternado ofrece buenos resultados, sin embargo, su grado de paralelización no es total ya que la solución se realiza por dominios pares e impares e involucra traslape de celdas.

Recomendaciones

1. Adicional a la metodología de Schwarz de descomposición de dominio, es posible implementar otras metodologías con mayor grado de paralelización; tal es el caso de los métodos de subestructuración para mejora del presente trabajo, método con el cual, el grado de paralelización se puede incrementar.

2. Implementar esquema de comunicaciones más eficiente y robusto, ya que la demanda de recursos aumenta con el tiempo, por lo que a través de técnicas de balance de carga se puede optimizar el desempeño del algoritmo.

3. Evaluar el algoritmo en equipos de cómputo con diferentes arquitecturas, para determinar el grado de portabilidad.

4. Extender la descomposición de dominio a dos direcciones y evaluar su rendimiento.

NOMENCLATURA

B Factor de volumen, m^3/m^3 (*rbl/sbl*)

B_o Factor de volumen del aceite, m^3/m^3 (*rbl/sbl*)

B_g Factor de volumen del gas, m^3/m^3 ($pie^3/MSCF$)

B_w Factor de volumen del agua, m^3/m^3 (*rbl/sbl*)

b_o Factor de encogimiento del aceite, $1/B_o$

b_g Factor de encogimiento del gas, $1/B_g$

b_w Factor de encogimiento del agua, $1/B_w$

c_r Compresibilidad de la roca, *1/pa* (*1/psi*)

D Profundidad, *m* (*pie*)

g Fuerza de gravedad, m/seg^2 (pie/seg^2)

G_w Factor geométrico del pozo, m^3 (pie^3)

H Altura, *m* (*pie*)

h Espesor de la formación, *m* (*pie*)

J Matriz jacobiana

k Permeabilidad absoluta, m^2 (*mD*)

k_e Permeabilidad absoluta, m^2 (*mD*)

k_f Permeabilidad de la fractura, m^2 (*mD*)

k_H Permeabilidad horizontal, m^2 (*mD*)

k_m Permeabilidad de la matriz, m^2 (*mD*)

k_r Permeabilidad relativa, *fracción*

k_{ro} Permeabilidad relativa al aceite, *fracción*

k_{rg} Permeabilidad relativa al gas, *fracción*

k_{rw} Permeabilidad relativa al agua, *fracción*

k_x Permeabilidad en dirección x, m^2 (*mD*)

k_y Permeabilidad dirección y, m^2 (*mD*)

l_x longitud en la dirección x, *m* (*pie*)

l_y longitud en la dirección y, *m* (*pie*)

l_z longitud en la dirección z, *m* (*pie*)

$p_{b,ijk}$ Presión del bloque *ijk*, *Pa* (*psi*)

p_o Presión del aceite, *Pa* (*psi*)

p_g Presión del gas, *Pa* (*psi*)

p_w Presión del agua, *Pa* (*psi*)

p_{wf} Presión de fondo fluyendo, *Pa* (*psi*)

$p_{wf,ref}$ Presión de fondo fluyendo de referencia, *Pa* (*psi*)

$p_{wf,sp}$ Presión de fondo fluyendo especificada, *Pa* (*psi*)

P_{cgo} Presión capilar gas aceite, *Pa* (*psi*)

P_{cwo} Presión capilar agua aceite, *Pa* (*psi*)

q_o Ritmo de producción/inyección de aceite, m^3/m^3s (pie^3/pie^3s)

q_g Ritmo de producción/inyección de gas, m^3/m^3s (pie^3/pie^3s)

q_w Ritmo de producción/inyección de agua, m^3/m^3s (pie^3/pie^3s)

q_{sc} Ritmo de producción/inyección a condiciones estándar, m^3/s (pie^3/s)

q_{spsc} Ritmo de producción/inyección a condiciones estándar especificado, m^3/s (pie^3/s)

$q_{well,ijk}$ Ritmo de producción/inyección del pozo en la celda *ijk*, m^3/s (pie^3/s)

$R_{o,ijk}$ Función de residuos del aceite en la celda ijk, m^3/m^3seg (pie^3/pie^3seg)

$R_{g,ijk}$ Función de residuos del gas en la celda ijk, m^3/m^3seg (pie^3/pie^3seg)

$R_{w,ijk}$ Función de residuos del agua en la celda ijk, m^3/m^3seg (pie^3/pie^3seg)

$\hat{R}_s$ Relación de solubilidad, m^3/m^3, *MSCF/bl*

r Distancia radial, *m* (*pie*)

r_o Radio equivalente de drene, *m* (*pie*)

r_w Radio del pozo, *m* (*pie*)

S_o Saturación de aceite, m^3/m^3 (pie^3/pie^3)

S_g Saturación de gas, m^3/m^3 (pie^3/pie^3)

S_w Saturación de agua, m^3/m^3 (pie^3/pie^3)
t Tiempo, *seg*
T_o Transmisibilidad del aceite, *1/Pa-seg* (*1/psi-seg*)
T_g Transmisibilidad del gas, *1/Pa-seg* (*1/psi-seg*)
T_w Transmisibilidad del agua, *1/Pa-seg* (*1/psi-seg*)
T_{omf} Transmisibilidad matriz fractura de aceite, *1/Pa-seg* (*1/psi-seg*)
T_{gmf} Transmisibilidad matriz fractura de gas, *1/Pa-seg* (*1/psi-seg*)
T_{wmf} Transmisibilidad matriz fractura de agua, *1/Pa-seg* (*1/psi-seg*)
U_f Vector de incógnitas de las fracturas
U_m Vector de incógnitas de la matriz
V_f Volumen de fluido, m^3 (pie^3)
V_p Volumen poroso, m^3 (pie^3)
V_r Volumen de roca, m^3 (pie^3)
V_t Volumen total, m^3 (pie^3)
WI Índice de productividad del pozo, m^3/*Pa-seg* (pie^3/*psi-seg*)

Letras griegas

α Factor geométrico, m^2/m^2 (pie^2/pie^2)
β_c Factor de conversión de unidades, adimensional
γ_o Gradiente de presión del aceite, *Pa/m* (*psi/pie*)
γ_g Gradiente de presión del gas, *Pa/m* (*psi/pie*)
γ_w Gradiente de presión del agua, *Pa/m* (*psi/pie*)
γ_{wb} Gradiente de presión del pozo, *Pa/m* (*psi/pie*)
Γ_1 Frontera de Ω_1 dentro de Ω_2
Γ_2 Frontera de Ω_2 dentro de Ω_1
Δt Incremento de tiempo, *seg*
Δx Incremento en la dirección *x, m* (*pie*)
Δy Incremento en la dirección *y, m* (*pie*)
Δ_t Operador diferencial de tiempo
ε Tolerancia
μ Viscosidad, *Pa seg* (*cp*)
μ_o Viscosidad del aceite, *Pa seg* (*cp*)
μ_g Viscosidad del gas, *Pa seg* (*cp*)
μ_w Viscosidad del agua, *Pa seg* (*cp*)
ρ_o Densidad del aceite, kg/m^3 (lb/pie^3)
ρ_g Densidad del gas, kg/m^3 (lb/pie^3)
ρ_w Densidad del agua, kg/m^3 (lb/pie^3)
σ Factor de forma matriz fractura por unidad de volumen de roca, $1/m^3m^2$ ($1/pie^3pie^2$)
$\hat{\tau}_{omf}$ Transferencia másica de aceite matriz fractura, kg/m^3seg (lb/pie^3seg)
$\hat{\tau}_{gmf}$ Transferencia másica de gas matriz fractura, kg/m^3seg (lb/pie^3seg)
$\hat{\tau}_{wmf}$ Transferencia másica de agua, kg/m^3seg (lb/pie^3seg)
τ_{omf} Transferencia volumétrica de aceite matriz fractura, m^3/m^3s (pie^3/pie^3seg)
τ_{gmf} Transferencia volumétrica de gas matriz fractura, m^3/m^3s (pie^3/pie^3seg)
τ_{wmf} Transferencia volumétrica de agua matriz fractura, m^3/m^3s (pie^3/pie^3seg)
φ Porosidad, m^3/m^3 (pie^3/pie^3)
φ_f Porosidad efectiva, m^3/m^3 (pie^3/pie^3)
Ω Dominio sin considerar fronteras
Ω_1 Subdominio 1 sin considerar fronteras
Ω_2 Subdominio 2 sin considerar fronteras

Subíndices y superíndices

ce Condiciones estándar
f Fractura

g	Fase gas
i	Posición de la malla en la dirección de *x* o *r*
j	Posición de la malla en la dirección de *y* o θ
k	Posición de la malla en la dirección de *z*
ijk	Celda ijk de la malla
m	Matriz
mf	Matriz-fractura
n	Nivel de tiempo anterior
n+1	Nivel de tiempo siguiente
o	Fase aceite
p	Fase aceite, gas, o agua (*o*,*g*,*w*)
ν	Nivel iterativo anterior
$\nu+1$	Nivel iterativo siguiente
w	Fase agua

BIBLIOGRAFÍA

1. Alonso, J.M. 1997. Programación de Aplicaciones Paralelas con MPI (Message Passing Interface). http://www.sc.ehu.es/acwmialj/edumat/mpi.pdf

2. Anguille, L., Killough, J.E., Li, T.M.C. y Toepfer, J.L. 1995. Static and Dynamic Load-Balancing Strategies for Parallel Reservoir Simulation. Artículo SPE 29102, presentado en SPE Symposium on Reservoir Simulation, San Antonio, Texas, febrero, 12-15. http://dx.doi.org/10.2118/29102-MS.

3. Apon, A., Buyya, R., Jin, H. et al. 2001. Cluster Computing in the Classroom: Topics, Guidelines, and Experiences. http://cloudbus.cis.unimelb.edu.au/papers/CC-Edu.pdf.

4. Arana-Ortiz, V.H. y Rodríguez, F. 1996. A Semi-Implicit Formulation for Compositional Simulation of Fractured Reservoir. Artículo SPE 36108, presentado en SPE Latin American and Caribbean Petroleum Engineering Conference, Puerto España, Trinidad y Tobago, abril 23-26. http://dx.doi.org/10.2118/36108-MS.

5. Babu, D.K., Odeh, A.S., Al-Khalifa, A.J. y McCann, R.C. 1991. The Relation Between Wellblock and Wellbore Pressures in Numerical Simulation of Horizontal Wells. *SPE Res Eng* **6** (3): 324-328. SPE-20161-PA. http://dx.doi.org/10.2118/20161-PA.

6. Barney, B. 2007. Introduction to Parallel Computing. https://computing.llnl.gov/tutorials/parallel_comp/.

7. Barua, J. y Horne, R. N. 1989. Improving the Performance of Parallel (and Serial) Reservoir Simulators. Artículo SPE 18408, presentado en SPE Reservoir Simulation Symposium, Houston, Texas, febrero 6-8. http://dx.doi.org/10.2118/18408-MS.

8. Branco, C.M. y Rodríguez, F. 1996. A Semi-Implicit Formulation for Compositional Reservoir Simulation. *SPE Advanced Technology Series* 4 (1): 171-177. SPE-27053-PA. http://dx.doi.org/10.2118/27053-PA.

9. Buyya, R., ed. 1999. *High Performance Cluster Computing: Architectures and Systems.* Upper Saddle River, New Jersey: Prentice Hall.

10. Byrd, J. 1997. A Basic UNIX Tutorial. http://fsl.fmrib.ox.ac.uk/fslcourse/unix_intro/.

11. Chien, M.C.H., Wasserman, M.L., Yardumian, H.E., Chung, E.Y., Nguyen, T. y Larson, J. 1987. The Use of Vectorization and Parallel Processing for Reservoir Simulation. Artículo SPE 16025, presentado en SPE Reservoir Simulation Symposium, San Antonio, Texas, febrero 1-4. http://dx.doi.org/10.2118/16025-MS.

12. Cismaşiu, I. 2002. Parallel Algorithms for Non-Conventional Finite Element Computations on Distributed Architectures. PhD thesis, Universidade Técnica de Lisboa, Instituto Superior Técnico, Lisboa (julio, 2002).

13. Coats, K.H. 1969. Use and Misuse of Reservoir Simulation Models. *J. Pet Tech* **21** (11): 1391-1398. SPE-2367-PA. http://dx.doi.org/10.2118/2367-PA.

14. Coats, K.H. 1980. An Equation of State Compositional Model. *SPE J.* **20** (5): 363-376. SPE-8284-PA. http://dx.doi.org/10.2118/8284-PA.

15. Coats, K.H. 1982. Reservoir Simulation: State of the Art. *J. Pet Tech* **34** (8): 1633-1642. SPE-10020-PA. http://dx.doi.org/10.2118/10020-PA.

16. Coats, K.H. 1989. Implicit Compositional Simulation of Single-Porosity and Dual-Porosity Reservoirs. Artículo SPE 18427, presentado en SPE Symposium Reservoir Simulation, Houston, Texas, febrero 6-8. http://dx.doi.org/10.2118/18427-MS.

17. Ding, Y. 1996. A Generalized 3D Well Model for Reservoir Simulation. *SPE J.* **1** (4): 437-450. SPE-30724-PA. http://dx.doi.org/10.2118/30724-PA.

18. Ding, Y. y Renard, G. 1994. A New Representation of Wells in Numerical Reservoir Simulation. *SPE Res Eng* **9** (2): 140-144. SPE-25248-PA. http://dx.doi.org/10.2118/25248-PA.

19. Ding, Y., Renard, G. y Weill, L. 1998. Representation of Wells in Numerical Reservoir Simulation. *SPE Res Eval & Eng* **1** (1): 18-23. SPE-29123-PA. http://dx.doi.org/10.2118/29123-PA.

20. Dogru, A.H., Sunaidi, H.A., Fung, L.S., Habiballah, W.A., Al-Zamel, N. y Li, K. G. 2002. A Parallel Reservoir Simulator for Large-Scale Reservoir Simulation. *SPE Res Eval & Eng* **1** (5): 11-23. SPE-75805-PA. http://dx.doi.org/10.2118/75805-PA.

21. Dongarra, J.J., Otto, S.W., Snir M. et al. 1995. An Introduction to the MPI Standard. Technical Report CS-95-274, University of Tennessee, Knoxville, Tennessee (enero 1995).

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

23. Foster, I. 1995. *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering.* Reading, Massachusetts: Addison Wesley.

24. Gilman, J.R. y Kazemi, H. 1983. Improvements in Simulation of Naturally Fractured Reservoirs. *SPE J.* **23** (4): 695-707. SPE-10511-PA. http://dx.doi.org/10.2118/10511-PA.

25. Golub, G. y Ortega J.M. 1993. *Scientific Computing an Introduction with Parallel Computing*. Boston, Massachusetts: Academic Press Inc.

26. Gropp, W. D. y Lusk E. 2001. Installation and User's Guide to MPICH, a Portable Implementation of MPI, Version 1.2.7p1 The ch_p4 Device for Workstation Networks. Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Illinois. http://www.atc.unican.es/manuales/MPI/mpich/mpichman-chp4.pdf.

27. Gropp, W., Lusk, E., Ashton, D. et al. 2008. MPICH2 Installer's Guide, Version 1.0.7. Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Illinois, (abril 2, 2008). http://vmg.pp.ua/books/MPI/mpich2-1.0.7-win32-ia32/mpich2-doc-install.pdf.

28. Gropp, W., Lusk, E., Ashton, D. et al. 2008. MPICH2 User's Guide (Version 1.0.7). Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Illinois, (abril 2, 2008). http://bluegrit.cs.umbc.edu/images/user.pdf.

29. Hansen, B.P. 1972. Structured Multi-programming. *Communications of the ACM* **15** (7): 574-578. http://www.matematicas.unam.mx/jloa/Articulos/p574-hansen.pdf.

30. Hansen, B.P. 1978. Distributed Processes: A Concurrent Programming Concept. *Communications of the ACM* **21** (11): 934-941. http://www.matematicas.unam.mx/jloa/Articulos/p934-hansen.pdf.

31. Hemanth-Kumar, K. y Young, L.C. 1996. Parallel Reservoir Simulator Computations. *SPE Comp App* **8** (2): 50-53. SPE-29104-PA. http://dx.doi.org/10.2118/29104-PA.

32. Hoare, C.A.R. 1978. Communicating Sequential Processes. *Communications of the ACM* **21** (8): 666-677. http://dx.doi.org/10.1145/359576.359585.

33. Holmes, J. A. 1983. Enhancements to the Strongly Coupled, Fully Implicit Well Model: Wellbore Crossflow Modeling and Collective Well Control. Artículo SPE 12259, presentado en SPE Reservoir Simulation Symposium, San Francisco, California, noviembre 15-18. http://dx.doi.org/10.2118/12259-MS.

34. Kaarstad, T., Froyen, J., Bjorstad, P. y Espedal, M. 1995. A Massively Parallel Reservoir Simulator. Artículo SPE 29139, presentado en SPE Reservoir Simulation Symposium, San Antonio, Texas, febrero 12-15. http://dx.doi.org/10.2118/29139-MS.

35. Kazemi, H., Merril, L.S. Jr., Porterfield, K.L. y Zeman, P.R. 1976. Numerical Simulation of Water-Oil Flow in Naturally Fractured Reservoir. *SPE J.* **16** (6): 317-326. SPE-5719-PA. http://dx.doi.org/10.2118/5719-PA.

36. Killough, J.E. 1990. Vector and Parallel Computing in Reservoir Simulation. En *Third International Forum on Reservoir Simulation, Baden, Austria, julio 23-27.*

37. Killough, J.E. 1993. Is Parallel Computing Ready for Reservoir Simulation?: A Critical Analysis of the State of the Art. Artículo SPE 26634, presentado en 68th Annual Technical Conference and Exhibition, Houston, Texas, octubre, 3-6. http://dx.doi.org/10.2118/26634-MS.

38. Liu, W., Cao, J., Mezzatesta, A. y Zhu, P. 2000. Parallel Reservoir Simulation on Shared and Distributed Memory System. Artículo SPE 64797, presentado en SPE International Oil and Gas Conference and Exhibition, Beijing, China, noviembre 7-10. http://dx.doi.org10.2118/64797-MS.

39. Mattax, C.C. y Dalton, R.L. 1990. *Reservoir Simulation*, Vol. 13. Richardson, Texas; Monograph Series, SPE.

40. Matthews, C.S. y Russell, D.G. 2004. *Pressure Buildup and Flow Tests in Wells*, Vol. 1. Richardson, Texas: Monograph Series, SPE.

41. Message Passing Interface Forum. 1994. MPI: A Message-Passing Interface Standard. Technical Report, Universidad de Tennessee, Knoxville, Tennessee (abril 1994).

42. MPICH. 2013. Homepage, www.mcs.anl.gov/mpi/mpich/download.html.

43. Mrosovsky, I. y Ridings, R.L. 1974. Two-Dimensional Radial Treatment of Wells Within a Three-Dimensional Reservoir Model. *SPE J.* **14** (2): 127-138. SPE-4286-PA. http://dx.doi.org/10.2118/4286-PA.

44. Nolen, J.S. 1990. Treatment of Wells in Reservoir Simulation. En *Third International Forum on Reservoir Simulation, Baden, Austria, julio 23-27*.

45. Pancake, C.M. 1996. Is Parallelism For You? *IEEE Computational Science & Engineering* **3** (2): 18-37. http://dx.doi.org/10.1109/99.503307.

46. Pacs Training Group. 2001. Introduction to MPI. University of Illinois, http://people.sc.fsu.edu/~jburkardt/pdf/mpi_course.pdf.

47. Peaceman, D.W. 1978. Interpretation of Well-Block Pressures in Numerical Reservoir Simulation. *SPE J.* **18** (3): 183-194. SPE-6893-PA. http://dx.doi.org/10.2118/6893-PA.

48. Peaceman, D.W. 1983. Interpretation of Well-Block Pressures in Numerical Reservoir Simulation With Nonsquare Grid Blocks and Anisotropic Permeability. *SPE J.* **23** (3): 531-543. SPE-10528-PA. http://dx.doi.org/10.2118/10528-PA.

49. Ponting, D.K., Foster, B.A., Naccache, P.F., Nicholas, M.O., Pollard, R.K., Rae, J., Banks, D. y Walsh, S.K. 1980. An Efficient Fully Implicit Simulator. *SPE J.* **23** (3): 544-552. SPE-11817-PA. http://dx.doi.org/10.2118/11817-PA.

50. Principles of Reservoir Simulation. 2011. Scientific Software Intercomp, Inc. (Advanced Technology for the Petroleum Industry).

51. Roosta, S.H. 1999. *Parallel Processing and Parallel Algorithms: Theory and Computation.* New York: Springer.

52. Rossen, R. H. 1977. Simulation of Naturally Fractured Reservoirs With Semi-Implicit Source Terms. *SPE J.* **17** (3): 201-210. SPE-5737-PA. http://dx.doi.org/10.2118/5737-PA.

53. Silva, F.A.B., Lopes, E.P., Aude, E.P.L. et al. 2003. Parallelizing Black Oil Reservoir Simulation Systems for SMP Machines. En *Proceedings of the 36th Annual Symposium on Simulation (ANSS '03). IEEE Computer Society, Washington, DC, EUA*, 224-230.

54. Smith, B.F., Bjørstad, P.E. y Gropp, W.D. 1996. *Domain Decomposition, Parallel Multilevel Methods for Elliptic Partial Differential Equations.* New York: Cambridge University Press.

55. Stonebank, M. 2001. UNIX Tutorial for Beginners, (http://www.ee.surrey.ac.uk/Teaching/Unix/.

56. Thomas, G.W. 1982. *Principles of Hydrocarbon Reservoir Simulation.* Boston, Massachusetts: International Human Resources Development Corporation.

57. Thomas, L.K., Dixon, T.N. y Pierson, N.G. 1983. Fractured Reservoir Simulation. *SPE J.* **23** (1): 42-54. SPE-9305-PA. http://dx.doi.org/10.2118/9305-PA.

58. Verdiere, S., Quettier, L., Samier, P. y Thompson, A**.** 1999. Applications of a Parallel Simulator to Industrial Test Cases. Artículo SPE 51887, presentado en SPE Reservoir Simulation Symposium, Houston, Texas, febrero 14-17. http://dx.doi.org/10.2118/51887-MS.

59. Warren, J.E. y Root, P.J. 1963. The Behavior of Naturally Fractured Reservoirs. *SPE J.* **3** (3): 245-255. SPE-426-PA. http://dx.doi.org/10.2118/426-PA.

60. Yang, Y., Dai, T., Han, Z. et al. 2005. The Parallel Strategy of a Large Scale Simulation About Ten Millions Nodes to Reservoir With Multiple Layers. *International Journal of Numerical Analysis and Modeling* **2** (supp): 177-182. http://www.math.ualberta.ca/ijnam/Volume-2-2005/Special/18.pdf.

61. Young, L.C. y Zarantonello, S.E. 1991. High Performance Vector Processing in Reservoir Simulation. En *Proceedings of the 1991 ACM/IEEE Conference on Supercomputing (Supercomputing '91). New York, NY, EUA,* 304-315. http://dx.doi.org/10.1145/125826.126000.

62. Zhang, K., Wu, Y.S., Ding, C., Pruess, K. y Elmroth, E. 2001. Parallel Computing Techniques for Large-Scale Reservoir Simulation of Multi-Component and Multiphase Fluid Flow. Artículo SPE 66343, presentado en SPE Reservoir Simulation Symposium, Houston, Texas, febrero 11-14. http://dx.doi.org/10.2118/66343-MS.

63. Zhiyuan, M., Fengjiang, J., Xiangming, X. y Jiachang, S. 1995. Simulation of Black Oil Reservoir on Distributed Memory Parallel Computers and Workstation Cluster. Artículo SPE 29937, presentado en International Meeting on Petroleum Engineering, Beijing, China noviembre, 14-17. http://dx.doi.org/10.2118/29937-MS.

64. Zhuang, X. y Zhu, J. 1994. Paralellizing a Reservoir Simulator Using MPI. En Proc. Scalable Parallel Libraries Conference, octubre 12-14, 165-174. Los Alamitos, California: IEEE Computer Society Press. http://dx.doi.org/10.1109/SPLC.1994.376993.

APÉNDICE A. ECUACIONES DE FLUJO MULTIFÁSICO EN MEDIOS POROSOS

Para obtener las ecuaciones de flujo multifásico en medios porosos, primero se muestra el desarrollo de la ecuación de continuidad para flujo de una fase de densidad ρ, con una velocidad v, en un medio de porosidad φ y permeabilidad k, y posteriormente se muestran las ecuaciones para el aceite, gas y agua. La formulación de las ecuaciones de flujo multifásico debe considerar la distribución espacial de cada componente como una función del tiempo. Considerando un volumen elemental representativo del medio poroso en coordenadas rectangulares con flujo en las tres direcciones, cuyas dimensiones son Δx, Δy y Δz, como se muestra en la **Fig. A.1**, se realiza el siguiente balance de materia:

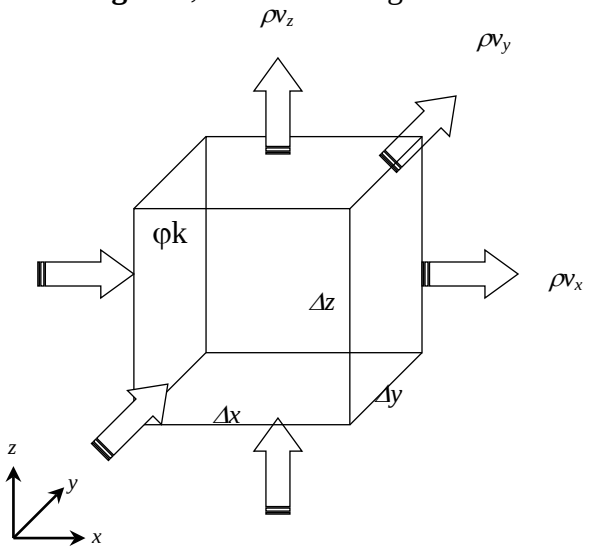


Fig. A.1 Volumen elemental representativo del medio poroso

$$\left\{\begin{matrix}Ritmo\ de\ entrada\\ de\ masa\ al\ elemento\end{matrix}\right\}-\left\{\begin{matrix}Ritmo\ de\ salida\\ de\ masa\ del\ elemento\end{matrix}\right\}\pm\left\{\begin{matrix}P\,roducción\\ o\ Inyección\end{matrix}\right\}=\left\{\begin{matrix}Ritmo\ de\\ acumulación\end{matrix}\right\} \tag{A.1}$$

Donde los términos del balance de materia son:

$$\left\{\begin{matrix}Ritmo\ de\ entrada\\ de\ masa\ al\ elemento\end{matrix}\right\}=\left(\rho v_x\right)_x\Delta y\Delta z+\left(\rho v_y\right)_y\Delta x\Delta z+\left(\rho v_z\right)_z\Delta x\Delta y\ , \tag{A.2}$$

$$\left\{\begin{matrix}Ritmo\ de\ salida\\ de\ masa\ del\,elemento\end{matrix}\right\}=\left(\rho v_x\right)_{x+\Delta x}\Delta y\Delta z+\left(\rho v_y\right)_{y+\Delta y}\Delta x\Delta z+\left(\rho v_z\right)_{z+\Delta z}\Delta x\Delta y\ , \tag{A.3}$$

$$\left\{\begin{matrix}P\,roducción\\ o\ Inyección\end{matrix}\right\}=\Delta x\Delta y\Delta z\rho q^*\ , \tag{A.4}$$

Donde q^* representa el gasto volumétrico por unidad de volumen de roca cuyas dimensiones son: L^3/TL^3.

$$\left\{\begin{matrix}Ritmo\ de \\ acumulación\end{matrix}\right\} = \Delta x \Delta y \Delta z \frac{\partial(\varphi\rho)}{\partial t} \ , \tag{A.5}$$

Sustituyendo los términos del balance y reagrupando, tenemos:

$$\begin{aligned}&\left[(\rho v_x)_x - (\rho v_x)_{x+\Delta x}\right]\Delta y \Delta z + \left[(\rho v_y)_y - (\rho v_y)_{y+\Delta y}\right]\Delta x \Delta z + \left[(\rho v_z)_z - (\rho v_z)_{z+\Delta z}\right]\Delta x \Delta y \\ &\pm \Delta x \Delta y \Delta z \rho q^* = \Delta x \Delta y \Delta z \frac{\partial(\varphi\rho)}{\partial t}\end{aligned} \tag{A.6}$$

Si dividimos la expresión anterior entre el volumen total del elemento representado como $\Delta x \Delta y \Delta z$, tenemos:

$$\left(\frac{\left[(\rho v_x)_x - (\rho v_x)_{x+\Delta x}\right]}{\Delta x}\right) + \left(\frac{\left[(\rho v_y)_y - (\rho v_y)_{y+\Delta y}\right]}{\Delta y}\right) + \left(\frac{\left[(\rho v_z)_z - (\rho v_z)_{z+\Delta z}\right]}{\Delta z}\right) \pm q^* \rho = \frac{\partial(\varphi\rho)}{\partial t} \ , \tag{A.7}$$

Sacando el límite cuando $\Delta x \to 0$ y $\Delta y \to 0$, y además, aplicando la definición de derivada, se obtiene:

$$-\frac{\partial(\rho v_x)}{\partial x} - \frac{\partial(\rho v_y)}{\partial y} - \frac{\partial(\rho v_z)}{\partial z} \pm q^* \rho = \frac{\partial(\varphi\rho)}{\partial t} \ , \tag{A.8}$$

O bien,

$$\frac{\partial(\rho v_x)}{\partial x} + \frac{\partial(\rho v_y)}{\partial y} + \frac{\partial(\rho v_z)}{\partial z} \pm q^* \rho = -\frac{\partial(\varphi\rho)}{\partial t} \ , \tag{A.9}$$

Es la ecuación de continuidad en coordenadas cartesianas. Por otro lado, sabemos que el operador nabla se define como:

$$\nabla = \frac{\partial}{\partial x}\hat{i} + \frac{\partial}{\partial y}\hat{j} + \frac{\partial}{\partial z}\hat{k} \ , \tag{A.10}$$

Por lo que la ecuación de continuidad escrita en forma vectorial, resulta:

$$\nabla \bullet (\rho v) \pm q^* \rho = -\frac{\partial(\varphi\rho)}{\partial t} \ , \tag{A.11}$$

La velocidad del fluido para flujo laminar en el yacimiento, podemos expresarla mediante la ecuación de Darcy:

$$v = -\frac{k}{\mu}\left(\nabla p - \gamma \nabla D\right), \tag{A.12}$$

Y la densidad del fluido relacionarla por medio de la siguiente ecuación de estado:

$$\rho = \frac{\rho_{ce}}{B}, \tag{A.13}$$

Donde B es el factor de volumen de la fase y ρ_{ce} es la densidad del fluido a condiciones estándar. Sustituyendo estos valores en la ecuación (A.11), la ecuación de flujo queda como:

$$\nabla \bullet \left(\frac{\rho_{ce}}{B} \frac{k}{\mu}\left(\nabla p - \gamma \nabla D\right) \right) \pm q^* \frac{\rho_{ce}}{B} = \frac{\partial}{\partial t}\left(\varphi \frac{\rho_{ce}}{B} \right), \tag{A.14}$$

Dado que ρ_{ce} es una constante, e introduciendo la definición del factor de encogimiento $b = 1/B$ se puede escribir la ecuación anterior como:

$$\nabla \bullet \left(\frac{b}{\mu} k\left(\nabla p - \gamma \nabla D\right) \right) \pm b q^* = \frac{\partial}{\partial t}\left(\varphi b\right), \tag{A.15}$$

Para el caso de flujo multifásico realizamos un desarrollo similar al mencionado para las fases aceite, gas y agua, para esto consideramos las propiedades da cada una de las fases, así como las saturaciones y permeabilidades relativas, lo que nos da para el aceite:

$$\nabla \cdot \left(\frac{b_o k k_{ro}}{\mu_o}\left(\nabla p_o - \gamma_o \nabla D\right) \right) + b_o q_o = \frac{\partial}{\partial t}\left(\phi b_o S_o\right), \tag{A.16}$$

Para el gas:

$$\nabla \cdot \left(\hat{R}_s b_o \frac{k k_{ro}}{\mu_o}\left(\nabla p_o - \gamma_o \nabla D\right) + b_g \frac{k k_{rg}}{\mu_g}\left(\nabla p_g - \gamma_g \nabla D\right) \right) + \hat{R}_s b_o q_o + b_g q_g = \frac{\partial}{\partial t}\left[\phi\left(\hat{R}_s b_o S_o + b_g S_g\right)\right] \tag{A.17}$$

Donde consideramos el gas libre y el gas disuelto en el aceite para la realización del balance de materia. Finalmente para el agua tenemos:

$$\nabla \cdot \left(\frac{k k_{rw}}{B_w \mu_w} \left(\nabla p_w - \gamma_w \nabla D \right) \right) + b_w q_w = \frac{\partial}{\partial t} \left(\phi b_w S_w \right) , \tag{A.18}$$