DISEÑO E IMPLEMENTACION DE LA VERSION PARALELA DEL PROGRAMA PARA EL MODELAMIENTO DE PROPAGACION DE ONDAS ELASTICAS 2D EN CUERPOS SÓLIDOS USANDO EL MÉTODO DE DIFERENCIAS FINITAS ECOELAS2D

DIANA CONSUELO HORTUA BAYONA

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERIAS FÍSICO- MECÁNICAS

ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

2006

DISEÑO E IMPLEMENTACION DE LA VERSION PARALELA DEL PROGRAMA PARA EL MODELAMIENTO DE PROPAGACION DE ONDAS ELASTICAS 2D EN CUERPOS SÓLIDOS USANDO EL MÉTODO DE DIFERENCIAS FINITAS ECOELAS2D

DIANA CONSUELO HORTUA BAYONA

Trabajo presentado como requisito para optar por el título de Ingeniero de Sistemas

Director:

Henry Lamos Díaz

Codirector:

Víctor Martínez Abaunza

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS

ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

2006

A Dios, mi familia y Erick

AGRADECIMIENTOS

La autora expresa sus agradecimientos a:

El profesor Henry Lamos Díaz, profesor asociado de la escuela de Matemáticas y al Ing Victór Martínez Abaunza Profesor Cátedra, por su valiosa colaboración para la elaboración de este proyecto de grado.

A mis compañeros de lucha, señores Erick Ramón Meneses y Luis Guillermo Ortiz por su colaboración en la elaboración del cluster donde se realizó la codificación y pruebas.

Al centro de Calculo Científico CeCalULA de la Universidad de los Andes de Mérida, Venezuela, por la asesoría brindada a través de la pasantía de Computación de Alto Rendimento.

A mis padres y hermanas, por sus voces de aliento cuando más lo necesitaba.

TABLA DE CONTENIDO

TABLA DE CONTENIDO 6

LISTA DE FIGURAS 8

LISTA DE TABLAS 10

LISTA DE ANEXOS 11

INTRODUCCION 15

1. PRESENTACION DEL PROYECTO 17

1.1 PLANTEAMIENTO DEL PROBLEMA 17

1.2 OBJETIVOS 18

1.2.1 Objetivo General 18

1.2.2 Objetivos Específicos 19

2. MARCO TEÓRICO 20

2.1 DISEÑO DE ALGORITMOS PARALELOS 20

2.1.1 Particionamiento 22

2.1.3 Comunicación 25

2.1.3 Agrupación 28

2.1.4 Asignación 30

2.2 MÉTODO DE DIFERENCIAS FINITAS 33

2.2.1 Descripción 34

2.2.2 Grilla de Diferencias finitas 35

2.2.3 Aproximación de diferencias finitas 37

2.3 MODELADO DE PROPAGACION DE ONDAS 38

2.2.1 Concepto de Onda 40

2.2.3 Principios De La Teoría De La Elasticidad 43

2.2.4 Ley De Hooke 46

2.2.5 Constantes elásticas en medios isotrópicos 47

3. METODOLOGIA 51

3.1 ESTUDIO DE LA TEORIA DE COMPUTACION PARALELA: 52

3.2 ESTUDIO DE ECOELAS2D 1.0 52

3.3 DISEÑO DEL ALGORITMO PARALELO: 52

3.4 IMPLEMENTACION DEL ALGORITMO 53

3.5 PRUEBAS DE EFICIENCIA 53

4 DESARROLLO DEL SOFTWARE 54

4.1 RECOLECCION Y ANALISIS DE REQUISITOS 54

4.1.1 Entendimiento del Problema 54

4.1.2 Requerimientos iniciales 63

4.2 DISEÑO 64

4.2.1 Partición 64

4.2.2 Comunicación 64

4.2.3 Agrupación 66

4.2.4 Asignación 67

4.3 IMPLEMENTACIÓN 67

4.3.1 Lenguaje de Programación 67

4.3.1 Formato datos de entrada 68

4.3.2 Implementación del algoritmo 69

5. PRUEBAS 79

5.1 LEY DE AMDAHL 79

5.2 COMUNICACIÓN Y REQUERIMIENTOS DE MEMORIA 80

5.3 EJECUCION DEL ALGORITMO 80

6. RESULTADOS 81

6.1 DEMO 1 81

6.1.1 Aceleración 82

6.2 DEMO 2 84

6.2.1 Aceleración 85

6.3 COMUNICACIÓN Y REQUERIMIENTOS DE MEMORIA 87

7. CONCLUSIONES 89

8. RECOMENDACIONES 91

BIBLIOGRAFIA 92

ANEXOS 94

LISTA DE FIGURAS

Figura 1 Etapas Diseño Algoritmos Paralelos **22**

Figura 2 *Descomposición de dominio para un problema que involucra una grilla de tres dimensiones.* **24**

Figura 3 Estructura de tarea para un ejemplo de búsqueda. **25**

Figura 4 Tarea y estructura del canal en una grilla de dos dimensiones que aplica el método de diferencias finitas con 5 puntos. **27**

Figura 5 Un algoritmo centralizado de suma que usa una tarea lider S para sumar N números distribuidos a lo largo de n tareas **27**

Figura 6 Ejemplos de agrupación: **28**

Figura 7 Granularidad Fina **29**

Figura 8 Asignación de un problema de grilla **31**

Figura 9 Método de Diferencias Finitas Explícito **34**

Figura 10 Método de diferencias finitas implícito **35**

Figura 11 Dominio de la solución y grilla de diferencias finitas **36**

Figura 12 Modelo geológico en dos dimensiones. **39**

Figura 13. Onda longitudinal **42**

Figura 14. Onda cortante **43**

Figura 15 La onda S y sus componentes SV y SH. **43**

Figura 16 Metodología **51**

Figura 17 Esquema de la grilla de diferencias finitas y plantilla espacial para la actualización de la velocidad. **56**

Figura 19. Fronteras Disipativas con superficie libre **60**

Figura 25 Distribución de las tareas . **71**

Figura 26 Estructura del modelo, velocidades p, s y la densidad. **83**

Figura 28 Tiempos totales para el demo 1 **84**

Figura 30. La aceleración para el demo2 **86**

Figura 31 Tiempos de ejecución para el demo 2 **86**

Figura 32 . Cantidad de datos que se intercambian en cada paso de tiempo por los procesadores. **87**

LISTA DE TABLAS

Tabla 1 Descripción Etapas Diseño Algoritmos Paralelos **21**

Tabla 2 Entradas del programa **62**

Tabla 3 Librerías y programas que componen a Ecoelas **70**

Tabla 4 Funciones programa principal **72**

Tabla 5 Variables creadas para la paralelización **73**

Tabla 6 Cambios Realizados librería Taper **74**

Tabla 7 Cambios Realizados A librería Higdon **76**

Tabla 8 Cambios Realizados librería Derivatives **78**

TABLA DE ANEXOS

ANEXO A Artículo Presentado en el congreso latinoameriacano de geofísica Bogotá **95**

ANEXO B Manual de Usuario **104**

RESUMEN

TITULO: Diseño e Implementación de la versión paralela del software para el modelamiento de propagación de las ondas elásticas 2d en cuerpos sólidos usando el método de diferencias finitas EcoElas2D[*]

AUTOR: Diana Consuelo Hortúa Bayona[**]

PALABRAS CLAVE: Modelamiento, Sismología Aplicada, Computación Paralela, Diferencias Finitas, MPI.

DESCRIPCIÓN:

El poder de cómputo ha avanzado a un estado donde podemos comenzar a realizar simulaciones cada vez más reales. Así, para modelar la propagación de ondas sísmicas en medios complejos se creó el software EcoElas2D, que utiliza el método de diferencias finitas para simular la propagación de ondas elásticas en dos dimensiones (2D). Sin embargo, en plataformas seriales los cálculos todavía se limitan a los tamaños pequeños en caso que el modelo exija una grilla muy fina, y/o un pequeño intervalo de tiempo. Este es un proceso típico de alta exigencia en CPU. La paralelización, una opción que permite incrementar el desempeño de la computación, se aplicó a un prototipo de este software.

Para hacer uso del poder de cómputo de los clusters, Ecoelas2D distribuye el trabajo sobre cada uno de los procesadores que lo componen. Usando el Interfaz de paso de mensajes (MPI) para la comunicación entre los procesadores, los tiempos de ejecución se reducen, así como los tamaños del modelo, aún para grillas pequeñas, se puede aumentar considerablemente. Una comparación de desempeño permite vislumbrar las ventajas de esta técnica.

Adicionalmente existe una versión más avanzada de este software que permite tener en cuenta topografía abrupta y hacer variaciones en el tamaño de la grilla a través del volumen. Esto permite optimizar el tamaño de la grilla y por lo tanto reducir el costo computacional. Un futuro desarrollo de esta última versión de EcoElas2D es también proceder a una paralelización

[*] Trabajo de Grado

[**] Facultad de Ingeniería Físico Mecánicas. Escuela de Ingeniería de Sistemas e Informática
Director: Ph.D Henry Lamos Díaz

ABTRACT

TITLE: Design and Implementation of the parallel version of software for the modelling of propagation of the elastic waves 2d in solid bodies using the method of finite differences EcoElas2D[*]

AUTHOR: Diana Consuelo Hortúa Bayona[**]

KEYS WORDS: Modelling, Seismology Aplicaded **,** Parallel Computation, Difference Finites, MPI.

DESCRIPTION:

The calculation power has advanced to a state where we can begin to make more and more real simulations. Thus, to model the propagation of seismic waves in average complexes the software EcoElas2D was created, that uses the method of finite differences to simulate the propagation of elastic waves in two dimensions (2D). Nevertheless, in serials platforms the calculations still limit the small sizes in case the model demands one grid very fine, and/or a small time interval. This it is a typical process of high exigency in CPU. The paralelization, an option that allows to increase the performance of the computation, was applied to a prototype of this software.

In order to make use of the power of calculation of clusters, Ecoelas2D distributes the work on each one of the processors that compose it. Using the Interface of passage of messages (MPI) for the communication between the processors, the run times are reduced, as well as the sizes of the model, for grids still small, can be increased considerably. A comparison of performance allows to glimpse the advantages of this technique.

Additionally a version is existed one more outpost of this software that allows to consider steep topography and to make variations in the size of grid through volume. This allows to optimize the size of grid and therefore to reduce the computational cost. A future development of this last version of EcoElas2D is also to come to a paralelization .

[*] Trabajo de Grado

[**] Facultad de Ingeniería Físico Mecánicas. Escuela de Ingeniería de Sistemas e Informática
Director. Ph.D Henry Lamos Díaz

INTRODUCCION

Este trabajo hace parte de PETROSISMICA, convenio de cooperación tecnológica UIS-CP, N- 005.

El Instituto Colombiano de Petróleo ICP posee una herramienta llamada EcoeElas2D desde el año 2003, un Software que modela la propagación de la onda elástica en dos dimensiones, al que no se ha podido sacar mayor provecho, debido a que las simulaciones emplean una cantidad de tiempo considerable, haciendo que se los modelos que se pueden ejecutar, sean de un tamaño reducido.

En el año 2004, se realiza la compra de un cluster de 8 nodos, como una ayuda computacional a los trabajos que tienen una carga computacional grande. Es allí cuando surge la idea de aprovechar los recursos disponibles en el instituto, con el talento humano existente en la Universidad Industrial de Santander UIS, para realizar la versión paralela de la aplicación y así solucionar los inconvenientes que representa el rendimiento presentado por el software secuencial.

El primer paso, fue la definición del marco de trabajo, para lo cual se sostuvieron reuniones con el tutor del ICP Msc. Saúl Guevara, quien participó en el desarrollo de las 2 versiones secuenciales que existen de la herramienta. Fue así como se determino la versión sobre la cual se trabajaría, siendo la primera la más indicada, y se definió el objetivo a conseguir a través de la realización de este proyecto.

El segundo paso, consistió en la consecución de la información necesaria: teoría de computación paralela, el método de diferencias finitas (es el método numérico utilizado para desarrollar las ecuaciones diferenciales parciales del modelo), y sismología computacional. Para ello se consultó libros, revistas, artículos, Internet, etc, Como resultado de este paso, surgió el plan de trabajo, definición de objetivos específicosl

La descripción del proceso seguido para el desarrollo del algoritmo paralelo es el tema a abordar en este libro. En primer lugar se presenta el marco teórico que muestra la técnica de diseño de algoritmos paralelos que fue empleada para el desarrollo de este proyecto, el método de diferencias finitas y por último una fundamentación del modelado de la propagación de ondas, que incluye algunos conceptos de básicos de geofísica.

El capítulo 2 y 3 corresponden a la metodología y desarrollo del software, en este último se explica detalladamente cada una de las etapas necesarias para el desarrollo del software, y como se llevaron a cabo.

En el capítulo 4 y 5 corresponde a las pruebas realizadas para medir el mejoramiento respecto a la versión secuencial, a través de métodos propuestos para este tipo de pruebas.

En el capitulo 6 se exponen las conclusiones que obtuvieron a través del desarrollo de este proyecto de investigación

Y por último en el capitulo 7 se exponen algunas recomendaciones para darle continuidad a esta área de investigación dentro del grupo de Petrosísmica UIS-ICP

1. PRESENTACION DEL PROYECTO

1.1 PLANTEAMIENTO DEL PROBLEMA

Perforar la Tierra para extraer petróleo es un proceso caro y, en demasiadas ocasiones, infructuoso. Entre el 80 y el 90 por ciento de las perforaciones acaban en sedimentos inútiles en lugar de un yacimiento. Para excavar los pozos y poder extraer el líquido negro, las compañías petrolíferas se ayudan normalmente de en la sismología aplicada con la que se trazan el mapa de los estratos de la zona utilizando tests de ultrasonidos. A partir del patrón de capas obtenido, los expertos identifican los puntos donde las condiciones podrían favorecer la acumulación de petróleo.

El modelamiento sísmico permite a partir de la teoría de propagación de las ondas simular el comportamiento de una perturbación sísmica a través de un medio de características físicas conocidas. Esta es una herramienta de gran utilidad en el estudio de la sismología aplicada, en particular en el caso de la exploración de recursos naturales de la tierra por medio de sísmica, permite hacer pruebas de hipótesis acerca de las características geológicas y conocer los efectos de estas características.

Los modelos teóricos pueden tener una gran variedad y diferentes grados de exactitud y esto se refleja en los diferentes algoritmos existentes en el modelamiento sísmico. Uno de ellos, el trazamiento de rayos se basa en un modelo muy simplificado, pero que tiene grandes ventajas desde el punto de vista computacional. El método de Diferencias Finitas es una opción mucho más

exacta, ya que se basa en la ecuación de Onda completa, pero tiene limitaciones en su aplicación práctica debido a que requiere grandes recursos computacionales.

Teniendo en cuenta su exactitud, en el ICP de ECOPETROL S.A se implementó un algoritmo de diferencias finitas 2D para fines experimentales. Es un algoritmo elástico 2D, basado en el artículo de Levander (1988). El principal objetivo es utilizarlo para evaluar la respuesta sísmica en las zonas complejas de Colombia, en donde existe la necesidad de mejorar la imagen que se obtiene con el método sísmico. Estas zonas se caracterizan por la compleja estructura geológica en el subsuelo y la topografía rugosa en superficie. Este algoritmo es probablemente uno de los más indicado para este tipo de situación.

Con el objetivo de aumentar su eficiencia computacional, se planteó la necesidad y utilidad de la paralelización, cuyo resultado se presenta en este trabajo Esto se logra a través de la distribución el trabajo sobre cada uno de los procesadores que componen el cluster. Usando la Interfaz de paso de mensajes (MPI) para la comunicación entre los procesadores, los tiempos de ejecución se reducen, así como los tamaños del modelo aumentan considerablemente. Una comparación de desempeño permite vislumbrar las ventajas de esta técnica.

1.2 OBJETIVOS

1.2.1 Objetivo General

Diseñar e implementar la versión paralela del programa para el modelamiento de propagación de ondas elásticas 2d en cuerpos sólidos usando el método de

diferencias finitas EcoELas2D 1.0 propiedad del Instituto Colombiano de Petróleo ICP, a través de la librería de paso de mensajes MPI, para mejorar sus tiempos de ejecución.

1.3 Objetivos Específicos

- Aprender el Estándar de Interfase de paso de mensajes (MPI), y algunas técnicas de paralelización para la resolución de una ecuación diferencial parcial por el método de diferencias finitas.

- Conocer el software de modelado de la ecuación de onda elástica 2D EcoElas2D, a través de un seguimiento del código línea por línea, para reconocer sus partes paralelizables.

- Aplicar la metodología de diseño de algoritmos paralelos (Partición, Comunicación, Agrupación y Asignación) a Ecoelas2D, para obtener el esquema de la versión paralela.

- Realizar simulaciones numéricas a través de modelos geológicos, para medir la bondad tanto en eficiencia como en aceleración respecto a la versión secuencial.

2. MARCO TEÓRICO

2.1 DISEÑO DE ALGORITMOS PARALELOS

La mayoría de problemas de programación tiene soluciones paralelas. La mejor solución secuencial puede ser a menudo al peor solución paralela. La metodología en que se basó el para el diseño del algoritmo paralelo comprende 4 etapas: partición, comunicación, aglomeración y mapeo. Las 4 etapas son mostradas en la figura 1 y se muestran en la siguiente tabla.

ETAPA	DESCRIPCION
Particionamiento	La tarea computacional y los datos son divididos en pequeñas tareas. En esta etapa el número de procesadores es ignorado, la atención se centra en las oportunidades de paralelismo
Comunicación	La comunicación requerida para coordinar la ejecución de tareas se determina, y se escoge los estructuras apropiadas de comunicación.
Agrupación	Las tareas y estructuras de comunicación definidas en las dos etapas anteriores son evaluadas con respecto a los requerimientos de rendimiento y los costos de implementación. Si es necesario, las tareas son combinadas para aumentar

	el rendimiento y reducir la comunicación.
Asignación	Cada tarea es asignada a cada procesador de manera que se satisfagan las metas de máxima utilización de procesamiento y mínimos costos de comunicación

Tabla 1 Descripción Etapas Diseño Algoritmos Paralelos

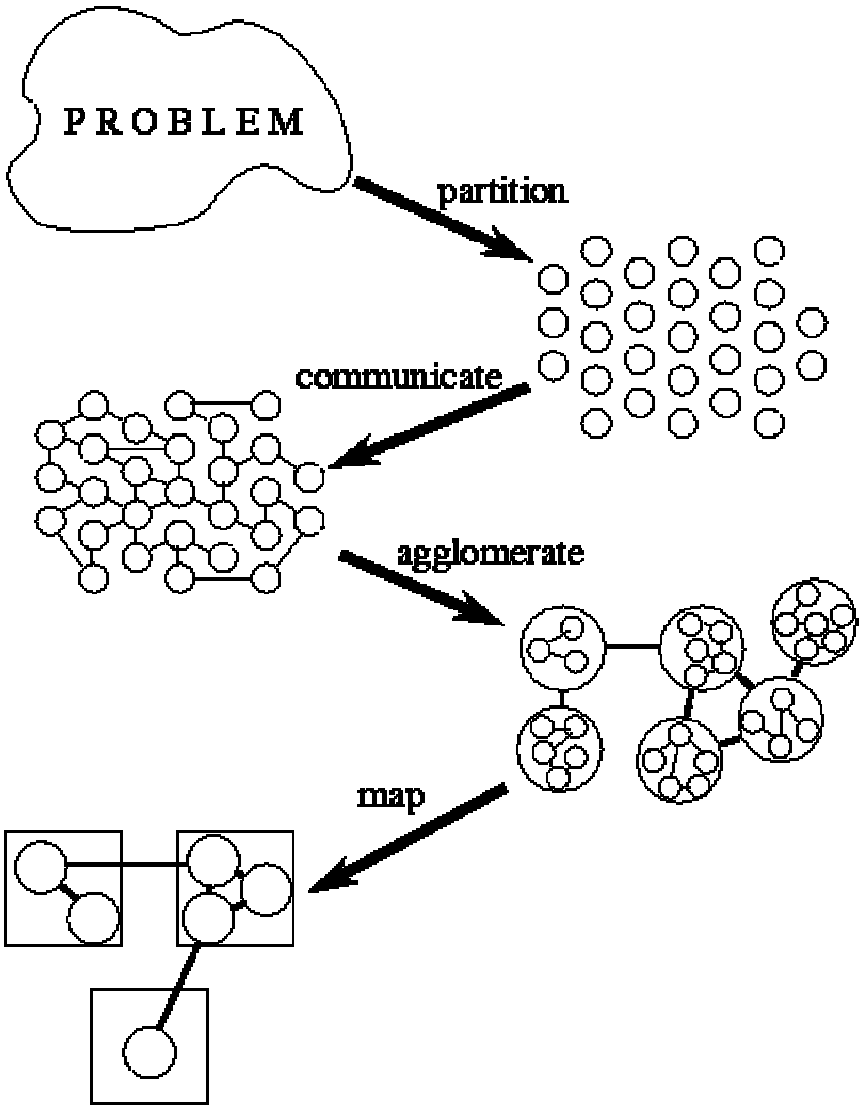


Figura 1. Etapas Diseño Algoritmos Paralelos (Tomada de Designing and Building Parallel Programs)

2.1.1 Particionamiento

La fase de partición se entende como buscar las oportunidades de ejecución paralela. La idea es definir una gran cantidad de tareas pequeñas para hacer una *granularidad fina*. Cuando la granularidad es fina es más fácil apilar las pequeñas tareas. Una descomposición de grano fino provee una excelente flexibilidad en

términos de aumentar el potencial de paralelización. En las siguientes fases, de acuerdo a la evaluación de requerimientos de comunicación, la arquitectura o software de desarrollo, se descartaran algunas oportunidades de paralelización.

Una buena paralelización divide en pequeñas piezas: se realiza división de instrucciones de máquina o división de los datos. Cuando se realiza una descomposición del los datos asociados con el problema se denomina *descomposición de dominio*. Otra técnica alternativa, primero descomponer el número de tareas, y después descomponer los datos se llama *descomposición funcional*.

1. **Descomposición de Dominio**

Con la descomposición de datos se da una aproximación a un problema de particionamiento, lo primero que se busca es la descomposición de los datos asociados al problema. Si es posible se busca dividir en pequeñas piezas del mismo tamaño. Después que la partición esta hecha, se asocia cada operación con los datos con los cuales va a operar. Esta partición produce un número de tareas, cada una restringida a los datos sobre los cuales va a operar. Una tarea puede requerir datos de diferentes tareas. En ese caso se necesita un canal para comunicar las diferentes tareas. Este requerimiento se analiza en la siguiente etapa.

Los datos pueden ser distribuidos en la entrada del problema, en la salida o mientras se realiza el proceso. Se puede realizar diferentes tipos de particiones, basado en diferentes estructuras de datos. Se sugiere que se empiece por la estructura de datos principal, o la que se tenga más acceso a lo largo del programa.

La figura 2 muestra la descomposición de un problema que tiene una grilla de 3 dimensiones. Las instrucciones son ejecutadas en cada punto de la grilla repetidamente. La descomposición en el eje x, y o z se puede realizar de la forma mas fácil. La descomposición más agresiva que en este caso sería, en cada punto de la grilla. Cada tarea mantendría en un estado varios valores asociados al punto de la grilla y es responsable por el cómputo que sea necesario para actualizar su estado.

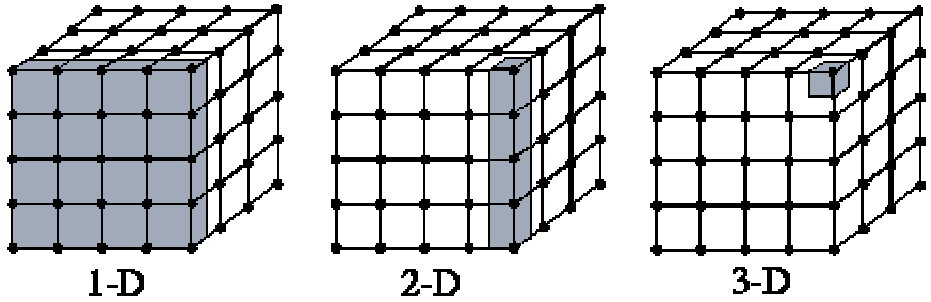


Figura 2 Descomposición de dominio para un problema que involucra una grilla de tres dimensiones. Es posible una descomposición de una, dos o tres dimensiones; en cada caso los datos asociados con una tarea es sombreado.

2. Descomposición Funcional

La descomposición funcional ofrece una alternativa diferente y complementaria. En esta aproximación la atención se centra en la computación que será ejecutada sin pensar en los datos que se utilizarán. Después de realizar la división en diferentes tareas, se procede a revisar los datos que se necesitarán para realizarla. Los requerimientos de datos pueden ser diferentes para cada tarea, en ese caso la partición ha sido completa. De lo contrario aparecen solapamientos de datos, en este caso será necesaria una cantidad de comunicación considerable. Esta es una razón por la cual se prefiere realizar una descomposición de datos.

Un ejemplo de cuando la descomposición funcional puede ser la más apropiada: una exploración en un árbol para la búsqueda del nodo “soluciones”. La descomposición del dominio seria obvia. Sin embargo una descomposición funcional de grano fino puede plantearse: inicialmente una tarea simple se crea en el nodo raíz del árbol, la tarea evalúa ese nodo, y si no es el nodo buscado crearía una nueva tarea (subárbol). Una nueva tarea sería creada cada vez que se expanda la búsqueda en el árbol (ver Figura 3)

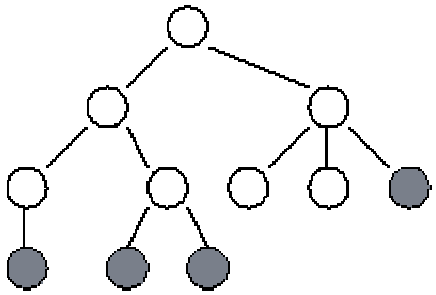


Figura 3 Estructura de tarea para un ejemplo de búsqueda. Cada circulo representa un modulo en el árbol de búsqueda y una llamada a la tarea nueva. Las líneas representan los canales donde retorna la solución.

2.1.3 Comunicación

Las tareas generadas en la partición son pensadas para ejecutarse concurrentemente, pero en general no son independientes. Las tareas generalmente requieren datos de otras. Los datos deben ser transmitidos de una tarea a otra para que pueda proseguir con su cómputo. Este flujo de información debe ser especificado en la etapa de comunicación.

Entonces surge la necesidad de conceptuar comunicación entre tareas, como un canal que las une, en el cual una tarea envía mensajes y otra la recibe. Entonces la comunicación asociada con un algoritmo puede ser especificada en 2 partes:

La primero define un canal que une, directa o indirectamente, tareas que requieren datos (consumidor) , con las tareas que poseen esos datos (productor)

En la segunda se especifica los mensajes que serán enviados y recibidos por ese canal.

La comunicación se puede clasificar como:

- **ü** Comunicación local cada tarea se comunica con un pequeño grupo de tareas “vecinas”, en contraste la comunicación global requiere que cada tarea se comunique con un gran número de tareas.
- **ü** Comunicación estructurada la tarea y sus vecinos forman una estructura regular como un árbol o una grilla; mientras que en la comunicación irregular la comunicación pueden ser grafos arbitrarios.
- **ü** En la comunicación sincrónica productores y consumidores ejecutan la tarea en forma coordinada, mientras que en la comunicación asincrónica el consumidor obtiene los datos sin ayuda del productor.

1. Comunicación local

Una estructura de comunicación local se obtiene cuando una operación necesita datos de un pequeño número de tareas Cuando este tipo de comunicación ocurre se debe crear un canal entre el consumidor y productor de tareas, e introducir operaciones de recibo y envío entre las tareas.

Por ejemplo, los requerimientos de comunicación asociados con un método numérico simple: El método de diferencias finitas de Jacobbi. La siguiente expresión se usa para actualizar cada uno de los puntos X_{ij} de la grilla X

$$X_{i.j}^{t+1} = \frac{4X_{i.j}^{t} + X_{i.-1.j}^{t} + X_{i+1.j}^{t} + X_{i\ j-1}^{t}\ X_{i\ j+1}^{t}}{8}$$

Esta actualización es aplicada repetidamente cuando $t=1,2,3, .. , n$ .La notación $X_{i.j}^{t}$ simboliza el valor del punto de la grilla $X_{i.j}$ en el paso de tiempo t.

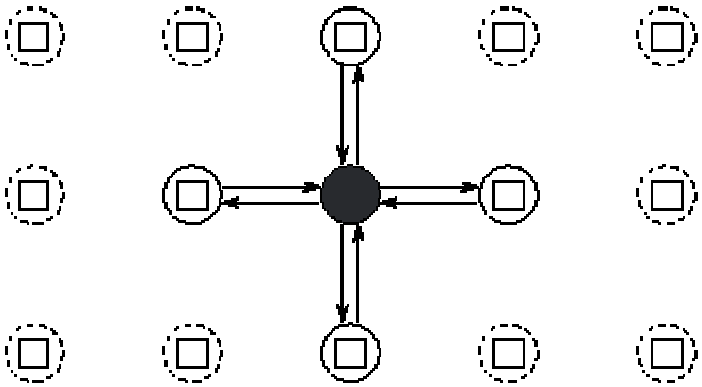


Figura 4 Tarea y estructura del canal en una grilla de dos dimensiones que aplica el método de diferencias finitas con 5 puntos. Solo son mostrados los canales en los cuales la tarea esta involucrada

2. Comunicación global

Una operación de comunicación global es aquella en la que varias tareas tienen que participar. Cuando cada operación es implementada, es difícil distinguir cada par de consumidores y productores.

Por ejemplo considere que se necesita recoger los resultados de N operaciones realizadas en N tareas usando el operador de la suma.

$$S=\sum_{i=0}^{n-1} X_i$$

Se asume una tarea líder que requiere el resultado de S. Desde el punto de vista local, se deben crear canales independientes en donde cada tarea 0,1,2,3,…, n envía su X_i a la tarea líder que debe acumular cada uno de los recibos en la variable S, como se muestra en la figura **-**.5

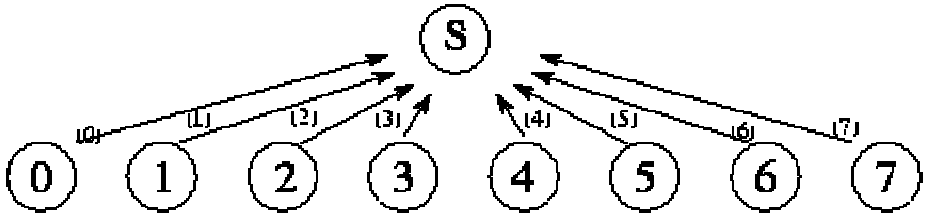


Figura 5 Un algoritmo centralizado de suma que usa una tarea lider S para sumar N números distribuidos a lo largo de n tareas. En este ejemplo N=7 cada uno de los canales esta etiquetado con el número del canal

2.1.3 Agrupación

En las primeras dos fases del proceso de diseño, se realiza la partición de la computación ejecutada en pequeñas tareas y se introduce la computación requerida por esas tareas. El algoritmo resultante es abstracto debido a que no esta orientado a una máquina paralela en especial. Este factor puede hacerlo ineficiente.

En esta tercera etapa, la agrupación, se mueve de lo abstracto a lo concreto. Se revisa si el algoritmo que se obtuvo en las 2 fases anteriores se adecúa a la maquina paralela donde se piensa implementar.

El número de tareas agrupadas en esta fase, que deben tender a la reducción, deben ser más grande que el número de procesadores a utilizar. Alternativamente, durante la etapa de agrupación se puede reducir el numero de tareas al numero de procesadores a utilizar (Figura 6)

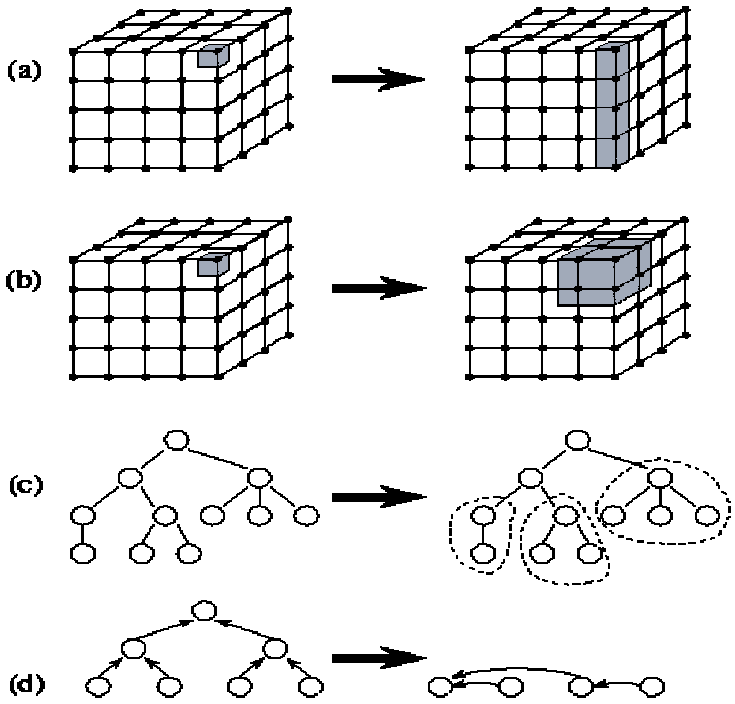


Figura 6 Ejemplos de agrupación: a) El tamaño del trabajo es incrementado para reducir la dimensión de descomposición de tres a dos. B) Las tareas adyacentes son combinadas para

lograr una descomposición de tres dimensiones de mayor granularidad. C) subárboles en una estructura divides y venceras son agrupados . d) Nodos de un árbol son combinados

Tres factores influyen a la hora de agrupar: granularidad, flexibilidad y Costos del Software.

1. Incrementar Granularidad

Una parte crítica que influencia el rendimiento de un algoritmo paralelo son los costos de comunicación. En la mayoría de maquinas paralelas, se tiene que parar la computación, para recibir y enviar mensajes.

Para resolver este problema se pude tomar dos caminos:

- Mejorar el rendimiento enviando menos datos.
- Disminuir el número de envíos realizados pero con la misma cantidad de datos

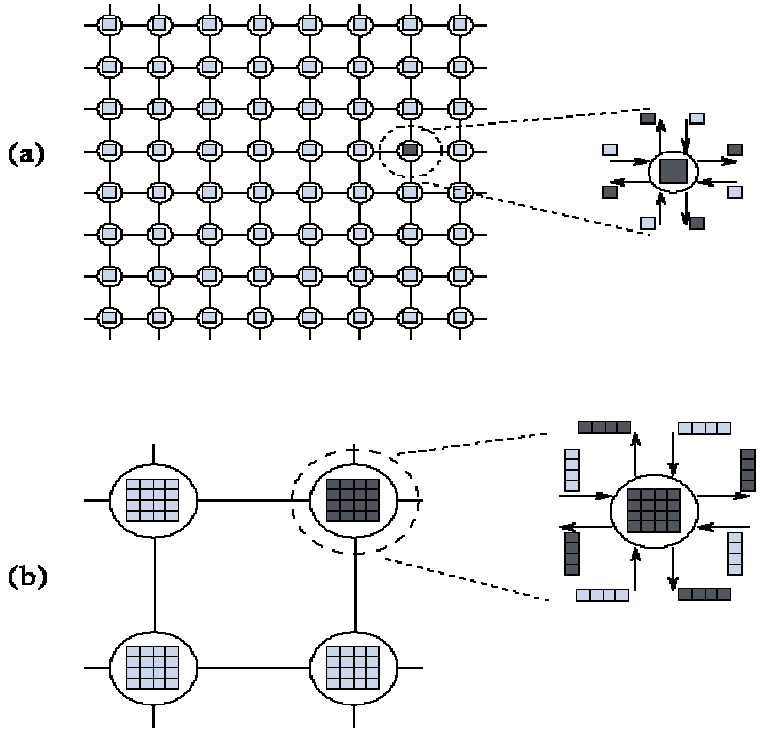


Figura 7 Granularidad Fina

La figura 7 muestra el costo de comunicación al hacer más grande la granularidad en un problema de diferencias finitas en 2D.
En la parte a) se muestra una grilla de 8*8 que es particionada en 8*8=64 tareas, cada una responsable por un punto, y hay un total de 64*4=256 comunicaciones, 4 por cada tarea.
Mientras que en la parte b) la grilla es dividida en 2*2=4 tareas cada una responsable por 16 puntos, y un total de 4*4=16 comunicaciones, 16*4=64 valores transferidos por tarea.

2. Preservar la Flexibilidad

La habilidad de crear tareas variables es crítica si se quiere que el programa sea escalable y portable. Esta flexibilidad es útil para adecuar el código para un computador en particular.
Crear mas tareas que procesadores permite una gran variedad de opciones en la etapa de asignación, permitiendo un mejor balanceo de carga.

3. Reduciendo costos de Ingeniería del Software

Una consideración adicional, cuando existen códigos secuenciales, es el costo relativo asociado con las diferentes técnicas de partición. Desde este punto de vista se puede escoger no realizar cambios significativos. Por ejemplo, si el código opera un grid multidimensional, puede ser ventajoso preservar las particiones alrededor de una dimensión, si este posee rutinas, que no puedan ser cambiadas en el programa paralelo.

2.1.4 Asignación

La cuarta y ultima etapa del proceso de diseño es la asignación, donde se especifica donde se va a ejecutar cada tarea. En esta clase de computadores, se

requiere, que el grupo de tareas y requerimientos de comunicación se encuentren bien especificados en el algoritmo paralelo; sistemas operativos o mecanismos de hardware, puede ayudar a asignar las tareas ejecutables en los procesadores disponibles. Desafortunadamente, los mecanismos de mapeo deben ser realizados para poder ser ejecutados en computadores paralelos escalables. En general, la asignación es un problema difícil que debe ser asignado explícitamente cuando se diseña algoritmos paralelos.

La meta al desarrollar algoritmos paralelos es minimizar el tiempo total de ejecución. Se pueden usar dos estrategias para lograrlo:

- **ü** Asignar tareas que se pueden ejecutar concurrentemente en diferentes procesadores para evitar la concurrencia.
- **ü** Asignar tareas que se comuniquen frecuentemente en el mismo procesador para incrementar la comunicación entre procesadores.

Estas estrategias, algunas veces causan conflicto, produciendo un desbalanceo de carga. Además las limitaciones de recursos pueden restringir el número de tares asignadas a un procesador.

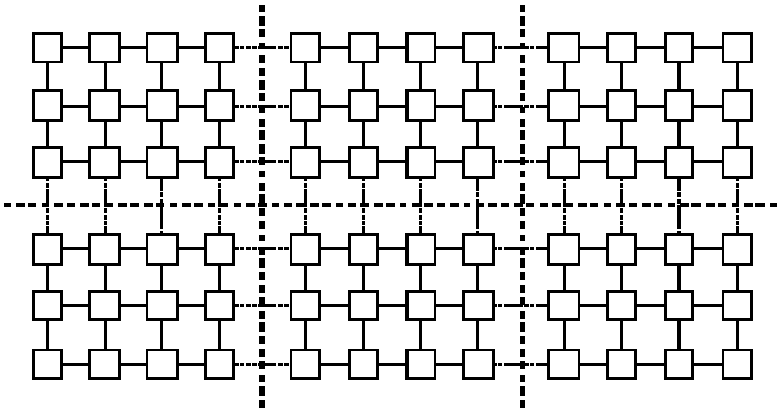


Figura 8 Asignación de un problema de grilla en el cual cada tarea ejecuta la misma cantidad de computación y comunicación solo con cuatro vecinos. Las lineas punteadas indican las fronteras de cada procesador

Muchos algoritmos desarrollados que usan descomposición de dominio usan la técnica de fijar tareas de igual tamaño, comunicación local estructurada y global. En estos casos una asignación eficiente es sencilla. Este mapeo de tareas minimiza la comunicación entre procesadores (Figura 8) también permite agrupar tareas en el mismo procesador, Esto produce un total de P tareas de grano grueso, una por procesador.

Una algoritmo más complejo que se base en la descomposición de dominio en el cual el trabajo por tarea y/o la comunicación no posea una estructura la agrupación y descomposición de dominio no es tan obvia para el programador. Es allí donde se utiliza algoritmos de *balanceo de carga* que buscan identificar estrategias eficientes de agrupación y asignación, usualmente técnicas heurísticas. El tiempo requerido para ejecutar estos algoritmos debe ser "pesado" con el fin de ganar en tiempo de ejecución.

Los problemas más complejos son aquellos en los cuales el número de tareas o la cantidad de comunicación o computación cambia dinámicamente durante el tiempo de ejecución. En ese caso el problema, se pueden usar la estrategia de *balanceo dinámico de carga*, con la cual un algoritmo es ejecutado periódicamente para determinar la nueva aglomeración y asignación.

Los algoritmos basados en descomposición funcional producen frecuentemente tareas que coordinan a otras tareas al inicio y final de la ejecución. En ese caso se pueden usar algoritmos de *programación de tareas*, que asignan tareas a un procesador que se encuentre ocioso.

2.2 MÉTODO DE DIFERENCIAS FINITAS

En la búsqueda de una descripción cualitativa de un determinado fenómeno físico, por lo general el ingeniero plantea un sistema de ecuaciones diferenciales ordinarias o parciales, válidas para determinada región (o dominio), e impone sobre dicho sistema condiciones de borde e iniciales apropiadas. En esta etapa, el modelo matemático está completo, y es aquí donde aparece la mayor dificultad, dado que solamente la forma más simple de ecuaciones, con fronteras geométricamente triviales es capaz de ser resuelta en forma exacta con los métodos matemáticos disponibles.

Las ecuaciones diferenciales ordinarias con coeficientes constantes son uno de los pocos ejemplos para los cuales se dispone de procedimientos matemáticos clásicos de solución.

Con el fin de evitar tales dificultades y lograr resolver el problema con la ayuda de computadoras, es necesario presentar el problema de una manera puramente algebraica.

Mediante el proceso de discretización, el conjunto infinito de números que representan la función o funciones incógnitas en el continuo es reemplazado por un número finito de parámetros incógnita, y este proceso requiere alguna forma de aproximación.

Entre las diferentes formas de discretización posibles (elementos finitos, volúmenes finitos, etc.), una de las más simples es mediante el Método de Diferencias Finitas.

2.2.1 Descripción

El objetivo de un método de de diferencias finitas para resolver una ecuación diferencial parcial es transformar el problema del cálculo a un problema de algebra :

- Discretizar el dominio físico dentro de un grid de diferencias
- Aproximar la derivadas parciales individuales en la ecuación diferencial parcial (PDE) por las aproximaciones algebraicas de diferencias finitas (FDAs)
- Sustituir las FDAs dentro de las PDE para obtener un ecuación algebraica de diferencias finitas (FDE)
- Resolver la FDEs

Los métodos de diferencias finitas en el cual al solución en el punto P en el nivel de tiempo n+1 dependente solo de sus puntos vecinos en el nivel de tiempo n . son llamados *métodos explícitos* porque la solución en cada punto es especificada explícitamente en términos de una solución conocida de los puntos vecinos en el nivel de tiempo n. Esta situación es ilustrada en la siguiente figura 9

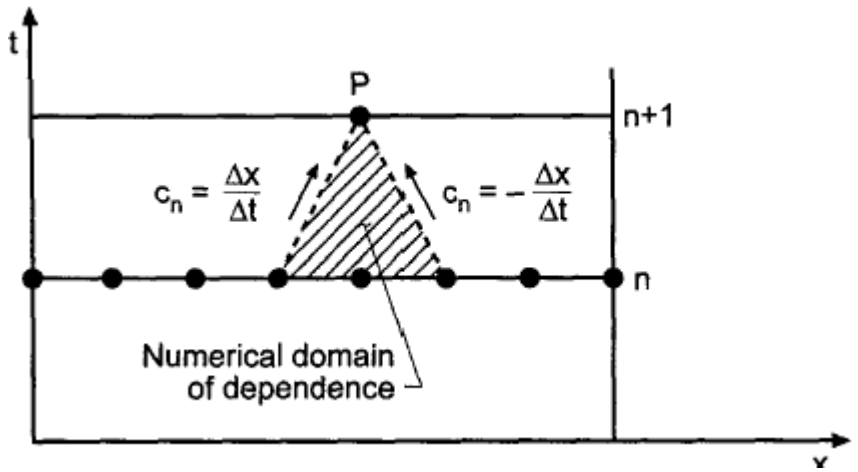


Figura 9 Método de Diferencias Finitas Explícito (Tomada de Numeral Methods for Engineers and Scientists 1998)

Los métodos de diferencias finitas en los cuales la solución en el punto P en el nivel de tiempo n+1 depende de la solución de los puntos vecinos en el nivel de tiempo n+1 además de la solución en el nivel de tiempo n se denominan *métodos implícitos* porque la solución en cada punto es especificada implícitamente, en términos de de una solución desconocida en los puntos vecinos del nivel de tiempo n+1. La situación es mostrada en la siguiente figura:

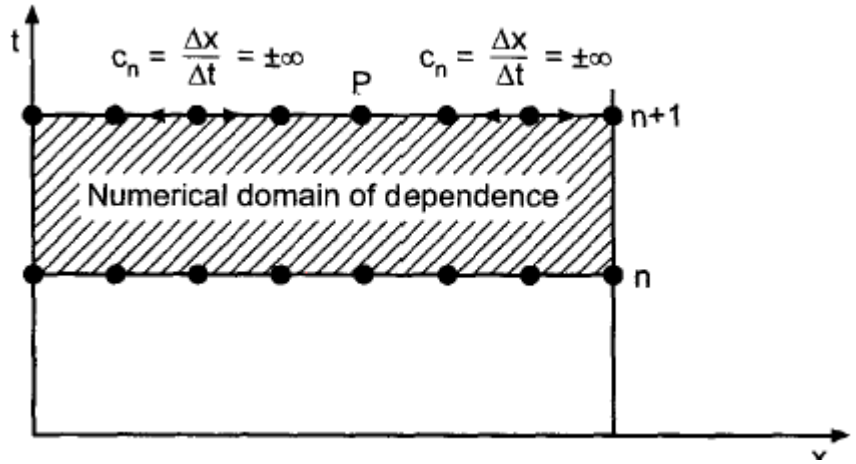


Figura 10 Método de diferencias finitas implícito (Tomada de Numeral Methods for Engineers and Scientists 1998)

2.2.2 Grilla de Diferencias finitas

El dominio de la solución *D(x,t)* en le espacio xt para un problema de una sola dimensión es mostrado en la siguiente figura. El dominio de solución debe ser cubierto por una cuadrícula, llamada grilla de diferencias finitas. Las intercepciones de las cuadrículas son los puntos de la grilla en los cuales se hallan con la solución de diferencias finitas de la ecuación diferencial parcial Las líneas de la grilla espaciales son perpendiculares al eje x, y están espaciadas uniformemente con una longitud Δx. La línea de la grilla esta espaciada con una longitud de Δt. Los subíndices i denotan las líneas de la grilla perpendiculares al eje x, y el subíndice n denota las líneas de la grilla perpendiculares al eje t Un punto de la grilla (i,n) corresponde a la localización (xi,t^n) en le dominio de solución

D(x,t). El numero total de líneas de grilla en x es llamado imax , y el numero total de pasos de tiempo es llamado nmax.

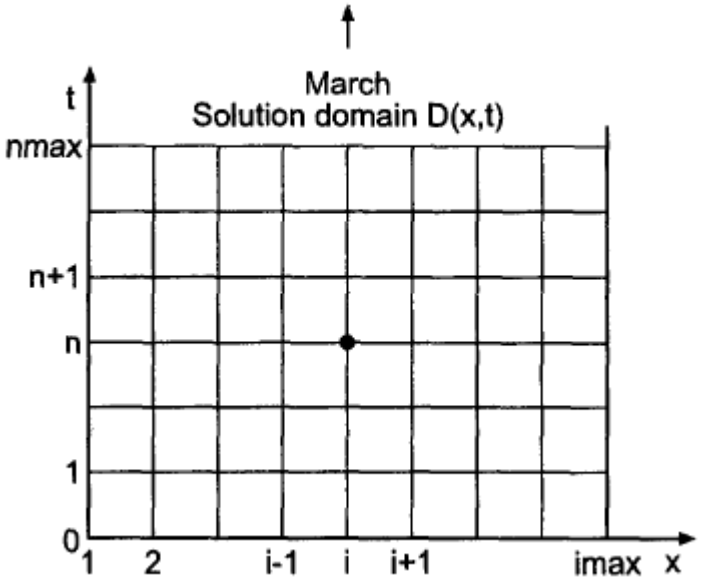


Figura 11 Dominio de la solución y grilla de diferencias finita**s** (Tomada de Numeral Methods for Engineers and Scientists 1998)

Los espacios físicos de dos dimensiones pueden ser cubiertos de manera similar por una grilla de planos perpendicular a los ejes de coordenadas, donde los subíndices i y j denotan los planos físicos de la grilla perpendiculares al eje x e y, respectivamente, y el subíndice n denota los planos de tiempo . Entonces el punto de la grilla (i,j,n) corresponde a la localización ($xi,yj.t^{n}$) en el dominio de solución D(x,y,t)

La variable dependiente en un punto de la grilla se escribe con la notación superíndice/subíndice de la siguiente manera;

$$f(x,t^n) = f^n$$

De la misma manera las derivadas:

$$\frac{\partial f(x_i, t^n)}{\partial t} = \frac{\partial f}{\partial t}\bigg|_i^n = f_t|_i^n \qquad \frac{\partial^2 f(x_i, t^n)}{\partial x^2} = \frac{\partial^2 f}{\partial x^2}\bigg|_i^n = f_{xx}|_i^n$$

2.2.3 Aproximación de diferencias finitas

Después de especificar la grilla de diferencias finitas, se procede a la obtención de las *aproximaciones de diferencias finitas.*

Estas aproximaciones se obtienen escribiendo las series de Taylor para la variable dependiente en uno o más puntos de la grilla usando uno o mas puntos de la grilla. Para funciones de una variable independiente, existen aproximaciones de varios tipos (por ejemplo, hacia delante, hacia atrás, y centrada) en varios órdenes (primer orden, segundo orden, etc) y desarrolladas para varias derivadas (primera derivada, segunda derivada, etc)

- **ü** Tipo de Aproximación

Para formar la aproximación de diferencias finitas, se debe tomar puntos vecinos de la grilla. Si estos puntos son tomados más puntos del posteriores que anteriores, se dice que es una aproximación hacia delante, por el contrario, si se tomas mas puntos anteriores que posteriores se dice que la aproximación es hacia atrás. Se dice que una aproximación es centrada, si se toman la misma cantidad de puntos anteriores como posteriores.

- **ü** Orden

El orden de una aproximación de diferencias finitas dice el término siguiente, sobre el cual fue truncada la serie de Taylor, en otras palabras indica el número de puntos en la grilla que fueron utilizados para realizar la aproximación.

2.3 MODELADO DE PROPAGACION DE ONDAS

En la industria petrolera, el proceso de modelado comienza con la toma de muestras en un terreno donde se tenga algún indicio de petróleo. Estas muestras son tanto de carácter físico (propiedades del las rocas), como pruebas electrónicas.

Las pruebas electrónicas consisten, realizar una perturbación en el medio(crear una ondícula) a través de instrumentos especializados, que se conocen como “fuente” y repartidos en zonas previamente definidas se encuentran otros instrumentos que recogen los tiempos de llegada, y velocidades de llegada de las ondas a su lugar de origen (llamados geófonos).

Los carácteristicas que se toman en estas muestras, son enviadas a los geólogos que con los datos tomados de campo realizan un modelo geológico que consiste en definir capas y las propiedades que se posee cada una de estas capas.

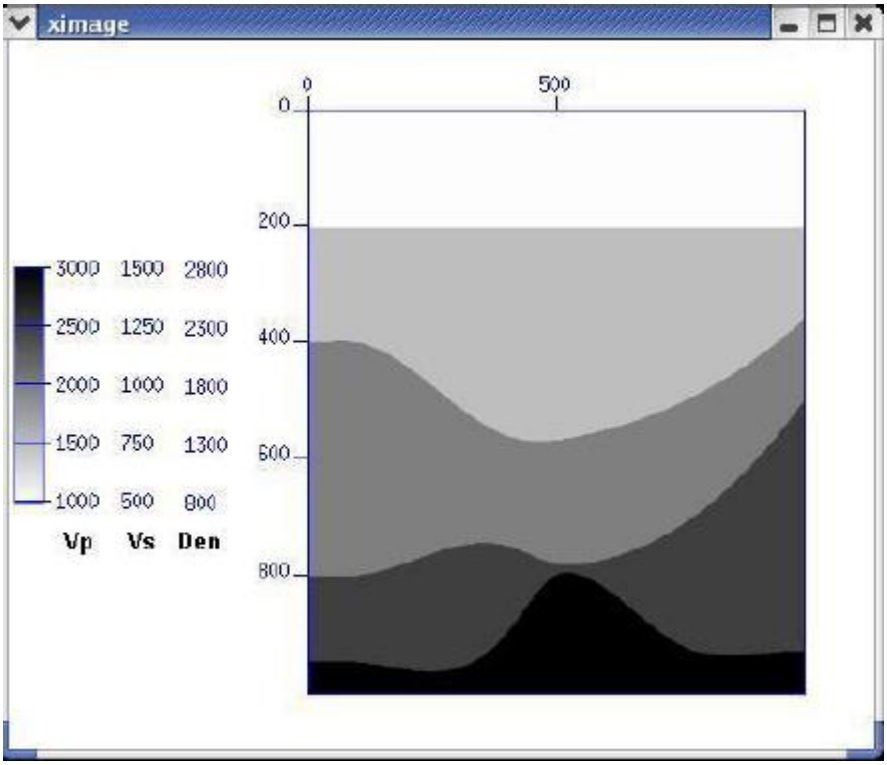


Figura 12 Modelo geológico en dos dimensiones. Al lado izquierdo se observa las propiedades del terreno

Después de este proceso, empieza el juego de prueba y error. Aplicando algunas de las técnicas de modelado para la propagación de ondas (también conocido como modelamiento sísmico), se comparan las resultados obtenidos en las pruebas de campo, con las arrojadas por el modelo computacional; en base a esta información se pueden "predecir" las características del todo el terreno y así buscar yacimientos o trampas de petróleo.

Para realizar el modelamiento sísmico, existen varios conceptos: Uno es el concepto de "rayo", de la óptica geométrica, una simplificación de gran utilidad, cuyas bases axiomáticas son los principios de Fermat y Huygens. Este concepto es aplicable para analizar trayectorias (con excepciones, como el fenómeno de la

difracción), como en el caso de la sísmica de refracción, en la que la propagación e interacción de las ondas con medios (suelo y roca) con propiedades variables se simplifica al hacer seguimiento a los rayos, que sufren los efectos de reflexión y refracción en las diferentes capas. Cuando se modela una perturbación a través del concepto de rayo, los datos que se obtienen son los tiempos de llegada del rayo y la amplitud del mismo.

El otro concepto fundamental es el que parte de la naturaleza real de la onda como propagación de una perturbación, necesario para explicar todos aquellos fenómenos en los cuales son determinantes las propiedades de la onda, por ejemplo el fenómeno de la difracción, transmisión de energía, interferencia, , la interacción de las ondas con propiedades del medio, etc. A través de este concepto se desarrolla la ecuación de la onda completa (que se verá más adelante) por medio de cualquier método numérico para resolver ecuaciones diferenciales parciales. Los datos que se obtienen son las velocidades de propagación de la onda.

El concepto explicado anteriormente, es en el que se basa este proyecto, utilizando las bases de la teoría elástica que se explicarán en las siguientes secciones.

2.2.1 Concepto de Onda

1. DEFINICION

El movimiento ondulatorio puede considerarse como un transporte de energía y cantidad de movimiento desde un punto del espacio a otro, sin transporte de materia.

Las ondas se clasifican en dos categorías: viajeras y estacionarias. En las primeras hay propagación de energía mientras que en las otras la energía asociada a la onda permanece confinada entre dos fronteras (p. ej. Gettys, 1991).

En la trayectoria de un frente de ondas se distinguen dos aspectos: 1) el movimiento de la onda a través del medio y, 2) el movimiento oscilatorio de las partículas del medio.

2. DESCRIPCIÓN DE LAS ONDAS

Los parámetros que se usan para describir una onda son: la frecuencia, $f = 1/T$, y la frecuencia angular, $\omega = 2\pi /T$, donde T es el periodo; y el número de onda, $k = 2\pi / \lambda$, donde λ es la longitud de la onda.

3. ONDAS ELÁSTICAS

En el modelamiento se generan dos tipos de ondas elásticas que se propagan a través del medio: las ondas de cuerpo o de volumen, y las ondas superficiales. La velocidad de propagación depende de la densidad del medio y de sus propiedades elásticas, el módulo de incompresibilidad y el módulo de rigidez .

Las ondas elásticas generan fuerzas y deformaciones que obedecen la teoría de la elasticidad (Ver seccion 2.3), en la cual los cuerpos sólidos tienen la propiedad de resistir cambios de tamaño o de forma, y de regresar a la condición no deformada cuando se eliminan las fuerzas externas.

4. ONDAS DE VOLUMEN

Primarias o de compresión (ondas P): son las que se propagan a mayor velocidad, por lo que a cualquier distancia del foco son registradas primero, de allá su nombre. Al propagarse hacen vibrar las partículas en el mismo sentido del tren de ondas, produciendo compresión y dilatación a su paso. Son conocidas también como ondas longitudinales.

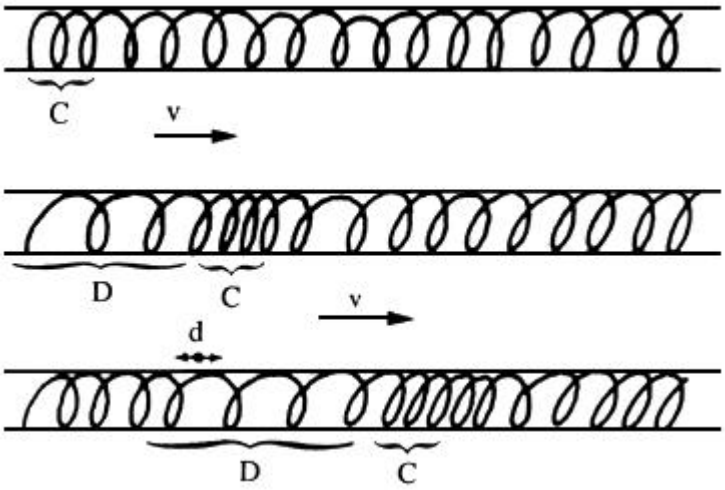


Figura 13. Onda longitudinal propagándose a lo largo de un resorte con velocidad v. C indica compresión y D indica dilatación. El desplazamiento de las partículas del resorte se produce en las direcciones indicadas por la flecha.

5. ONDAS SECUNDARIAS O S

Hacen vibrar las partículas en sentido perpendicular al de su propagación. Tienen velocidades menores que las ondas P. Si las partículas oscilan de arriba a abajo, la onda se llama SV, si las partículas oscilan en un plano horizontal se llaman SH (ver figura 3). También son conocidas como ondas transversales.

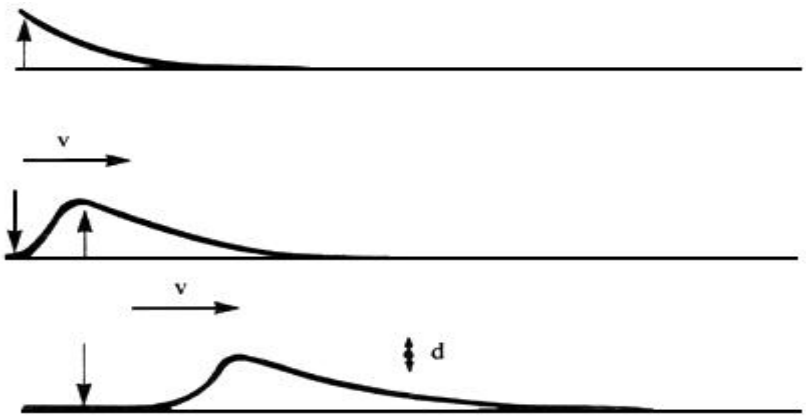


Figura 14. Onda cortante propagándose con velocidad va lo largo de una cuerda. El desplazamiento de las partículas de la cuerda se da en las direcciones indicadas por de la flecha.

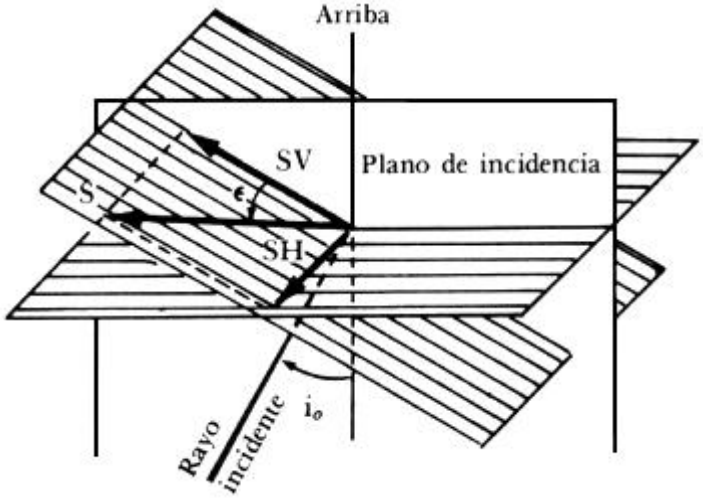


Figura 15 La onda S y sus componentes SV y SH**.**

2.2.3 Principios De La Teoría De La Elasticidad

Una perturbación sobre un medio elástico, en función del tiempo (p. ej. un sismo, el impacto de un meteorito, una explosión nuclear, el golpe de un martillo sobre el suelo) genera ondas elásticas. Estas perturbaciones producen cambios locales en esfuerzo y deformación.

Para entender la propagación de las ondas elásticas es necesario describir cinemáticamente la deformación del medio y las fuerzas resultantes – esfuerzos -. La relación entre deformación y esfuerzo está gobernada por las constantes elásticas.

La relación de estas perturbaciones con el tiempo lleva a la ecuación de las ondas elásticas.

1. ESFUERZO

Se define como la fuerza por unidad de área. Así-, cuando una fuerza es aplicada a la superficie exterior de un cuerpo, el esfuerzo es la relación de la fuerza en el área sobre la cual es aplicada:

Esfuerzo=Fuerza/área= F/A

Si la fuerza es perpendicular al área se llama esfuerzo normal de compresión. Cuando la fuerza es tangencial al área el esfuerzo se conoce como esfuerzo cortante o de cizalla.

Si se tiene un cuerpo de lados rectangulares de lado δ_x , δ_y y δ_z en cada uno de los sentidos *x , y* , y *z* de los ejes cartesianos coordenados, entonces los esfuerzos normales se definen como:

$$\tau_z = \frac{F_z}{\delta_y \delta_x} \qquad \tau_y = \frac{F_y}{\delta_z \delta_x} \qquad \tau_x = \frac{F_x}{\delta_y \delta_z}$$

2. DEFORMACIÓN

Cuando un cuerpo elástico está sujeto a esfuerzos ocurren cambios en la forma y en las dimensiones. Estos cambios se conocen como deformaciones. Así, la deformación se define como un cambio relativo en la dimensión (volumen) o forma un cuerpo.

Si se tiene un cubo de dimensiones X , Y y Z para cada uno de los ejes cartesianos x , y , y z , entonces se producirán dos tipos de deformaciones: normales y de cizalla.

La deformación primaria (o elemental) es la **deformación normal**. Según el eje cartesiano en que se produzca la fuerza se tendrá:

$$\varepsilon_x = \frac{\partial u}{\partial x} \quad \varepsilon_y = \frac{\partial v}{\partial y} \quad \varepsilon_z = \frac{\partial w}{\partial z}$$

Donde ∂u , ∂v y ∂w son los cambios en longitud de cada lado del cubo en los ejes coordenados x ,y , y z , respectivamente.

La **deformación de cizalla** se define como la combinación de deformaciones en los planos xy , xz o zy así:

$$\varepsilon_{xy} = \varepsilon_{yx} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}$$

$$\varepsilon_{yz} = \varepsilon_{zy} = \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}$$

$$\varepsilon_{xz} = \varepsilon_{zx} = \frac{\partial u}{\partial x} + \frac{\partial w}{\partial z}$$

Los cambios en las dimensiones dadas por las deformaciones normales resultan de los cambios en el volumen, cuando el cuerpo es deformado. El cambio en volumen por unidad de volumen es llamado **dilatación**, que puede representarse con la siguiente fórmula:

$$\Delta = \varepsilon_{xx} + \varepsilon_{yy} + \varepsilon_{xx}$$

2.2.4 Ley De Hooke

Para calcular las deformaciones cuando los esfuerzos son conocidos, se debe conocer la relación que existe entre el esfuerzo y la deformación. Cuando las deformaciones son pequeñas esta relación está dada por la Ley de Hooke, la cual establece que, dada una deformación, ésta es directamente proporcional al esfuerzo producido. Cuando existen varios esfuerzos, cada uno produce deformaciones, independiente de los otros esfuerzos, entonces el total de las deformaciones es la suma de las deformaciones individuales producidas por cada esfuerzo.

En medios isotrópicos es decir, cuando las propiedades o características del medio no varían, o no dependen de la dirección sobre la cual se aplican las fuerzas, la relación entre esfuerzo y deformación puede definirse de la siguiente forma:

$$\tau_{ii} = \lambda\Delta + 2\mu\varepsilon_{ii} \qquad \text{donde } i=x,y,z$$

$$\tau_{ij} = \mu\varepsilon_{ij} \qquad \text{donde } i=x,y,z \text{ para } i \neq j$$

Donde λ y μ son las constantes elásticas de Lamé; Δ es la dilatación y ε_{ii} y ε_{ij} las deformaciones, μ es una medida a la deformación de cortante y es conocido como el **Módulo de rigidez** al cortante o módulo de cizalla. Los líquidos no oponen resistencia a la cizalla, por lo tanto $\mu=0$.

2.2.5 Constantes elásticas en medios isotrópicos

Las constantes que describen el comportamiento elástico en un medio isotrópico son los módulos de Lamé y de rigidez. Existen tres módulos adicionales que permiten describir también el comportamiento elástico en términos de los dos primeros módulos, ellos son:

1. Módulo de elasticidad, E .
2. Módulo de incompresibilidad, K .
3. Cociente de Poisson, σ.

En la litosfera las rocas se aproximan a medios isotrópicos, es decir que no lo son completamente. Especialmente las rocas sedimentarias y metamórficas presentan anisotropías. Por ejemplo, las rocas sedimentarias presentan diferencias en sus propiedades si son medidas en planos paralelos o perpendiculares al plano de estratificación (p. ej. Briceño & Cuellar, 1991).

1. Módulo de elasticidad o de Young, E

Es la cantidad de esfuerzo por unidad de deformación.

E = Esfuerzo / Deformación

E = Fuerza por unidad de área / Cambio en longitud por unidad de longitud.

Considerando sólo esfuerzo normal el módulo elástico queda definido como:

$$E=\frac{\varepsilon_{ii}}{\sigma_{ii}}$$

Aplicando la Ley de Hooke se tiene:

$$E=\frac{\mu(3\lambda+2\mu)}{\lambda+\mu}$$

2. Módulo incompresibilidad, K

Es una medida de la resistencia de los materiales elásticos a la compresión, es decir, al cambio de volumen sin que varíe su forma. Si un cuerpo está sometido a esfuerzo de compresión en todas las direcciones, su volumen disminuirá una cantidad $_{ii}\varepsilon$. Así, el módulo de incompresibilidad es la relación entre el esfuerzo y el cambio unitario de volumen.

K = Esfuerzo / deformación

K = Presión / Cambio volumen por unidad de volumen.

Para definir el módulo de incompresibilidad, usualmente se supone que el cuerpo está sujeto solamente a la presión hidrostática, es decir:

$$\sigma_{xx}= \sigma_{yy}= \sigma_{zz}=-P$$

$$\sigma_{xy}= \sigma_{xz}= \sigma_{yz}=0$$

Entonces el módulo de incompresibilidad queda definido como:

$$K=-\frac{P}{\Delta}$$

El signo menos es insertado para que K sea positivo. Al sustituir según la Ley de Hooke se tiene:

$$K=\frac{3\lambda+2\mu}{3}$$

3. Cociente de Poisson, σ .

Es la relación entre las deformaciones unitarias transversal y longitudinal.

Para definirla asúmase que todos lo esfuerzos son cero excepto σ_{xx} . Entonces se tiene:

$$\sigma=-\frac{\varepsilon_{yy}}{\varepsilon_{ii}}=-\frac{\varepsilon_{zz}}{\varepsilon_{ii}}$$

donde el signo negativo es insertado para que el cociente sea positivo.

Al reemplazar según las ecuaciones de la Ley de Hooke se obtiene:

$$\sigma = \frac{\lambda}{2\,(\lambda + \mu)}$$

La relación de Poisson es una medida de la contracción lateral del material. En el caso de materiales elásticos varía entre 0 y 0,5. Como los líquidos no oponen resistencia a esfuerzo cortante, $\mu=0$, entonces $\sigma = ½$.

4. Ecuación de la onda en función de la distancia y el tiempo

Para describir la ecuación de la onda de un sólido elástico es necesario recurrir a la Ley de Newton.

$$\boldsymbol{F} = ma = \rho \frac{d^2 u}{d t^2}$$

Como la fuerza depende de la tasa de cambio espacial del esfuerzo, es obvio que si el esfuerzo es uniforme no hay fuerza. Por esta razón se puede recurrir entonces a la Ley de Hooke, que relaciona el esfuerzo en términos de la deformación.

Para una barra simple, donde el desplazamiento es *u(x,t)*

$$\boldsymbol{F} = ma = \rho \frac{d^2 u}{d t} = \frac{d}{\partial x}\left(E \frac{du}{dx} \right) = E \left(\frac{d^2 u}{dx^2} \right)$$

Como la velocidad de la onda longitudinal en una barra es:

$$Cp = \sqrt{\frac{E}{\rho}}$$

por lo tanto:

$$\frac{d^2 u}{dt^2} = Cp^2 \frac{d^2 u}{dx^2}$$

Que es lo mismo que:

$$\frac{d^2 u}{dx^2} = \frac{1}{Cp^2} \frac{d^2 u}{dt^2}$$

Esta es la ecuación general de una onda. La ecuación se puede satisfacer para cualquier onda en una sola dimensión que se propaga sin dispersión o sin variación de forma (e. g. Gettys, 1991).

5. Velocidades de las ondas elásticas

En un medio homogéneo la velocidad de las ondas elásticas depende de la densidad de masa del suelo ρ, y de los parámetros elásticos: Los parámetros de Lamé.

La velocidad de las ondas P y S estás dadas por las siguientes ecuaciones (p. ej. Slankisqui, 2003):

$$Cp = \sqrt{\frac{\lambda + 2\mu}{\rho}}$$

$$Cs = \sqrt{\frac{\mu}{\rho}}$$

3. METODOLOGIA

Uno de los puntos mas importantes para que cualquier proyecto sea desarrollado de forma exitosa, es la elección de una metodología adecuada, ya que ésta marcará el proceso de desarrollo del software.

La metodología que se seguirá en este proyecto, se muestra en la siguiente figura:

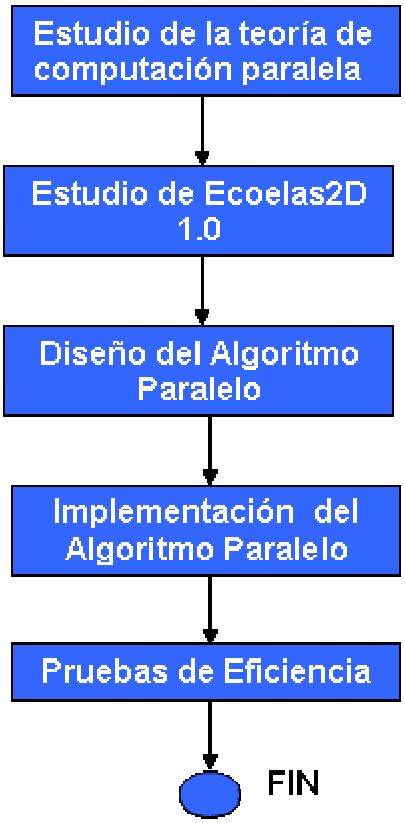


Figura 16 Metodología

3.1 ESTUDIO DE LA TEORIA DE COMPUTACION PARALELA:

En esta etapa se ordenará y estudiará la información recopilada sobre las técnicas de paralelización existentes, para tener los conocimientos necesarios para realizar el diseño del algoritmo paralelo.

3.2 ESTUDIO DE ECOELAS2D 1.0

En esta etapa se estudiará la aplicación secuencial existente: conocer sus características, el método numérico utilizado para realizar el modelamiento, condiciones de frontera e iniciales, etc. En esta etapa se realizaría el análisis y recolección de requisitos, debido a que la aplicación secuencial, apoyado con las orientaciones del tutor del ICP, brindarán las bases necesarias para realizar el diseño del algoritmo.

3.3 DISEÑO DEL ALGORITMO PARALELO:

El diseño en cualquier proyecto, es la fase mas importante en la construcción de un proyecto software, pues allí se definen su estructura, que es el soporte en la fase de codificación del sistema.

El diseño que se seguirá en este proyecto es el planteado por Ian Foster en su libro “Designing and Building Parallel Applications” en donde se plantean 4 fases:

- ü Partición
- ü Comunicación
- ü Agrupamiento
- ü Asignación

Para mayores detalles sobre las técnicas de diseño, puede referirse al capitulo 2 donde se explica cada una de las 4 fases.

3.4 IMPLEMENTACION DEL ALGORITMO

La implementación o construcción es la etapa en donde se traduce el diseño en un lenguaje de programación, es decir se expresa como código todas las operaciones y funciones del sistema.

3.5 PRUEBAS DE EFICIENCIA

Las pruebas de eficiencia se realizan sobre el algoritmo paralelo para “evaluar” la mejora respecto el algoritmo secuencial. Las medidas que se utilizarán en este proyecto es la aceleración medida a través de la Ley de Amdahl's[1],

[1] Amdahl G.The validity of the single processor approach to achieving large scale computing capabilities. :w

4. DESARROLLO DEL SOFTWARE

4.1 RECOLECCION Y ANALISIS DE REQUISITOS

4.1.1 Entendimiento del Problema

La primera fase de este proyecto fue entender el software secuencial que ya estaba desarrollado, y así lograr un entendimiento del problema que se estaba abordando, para ello se contó con la ayuda del tutor del ICP Msc. Saúl Ernesto Guevara. Para un mayor entendimiento del lector, a continuación se hace una breve descripción de las características de la aplicación secuencial.

1. Características

EcoElas2D esta basado en la aproximación en diferencias finitas de la ecuación de onda elástica presentada por Alan Levander en el artículo ***"Fourth-order finite-difference P-SV seismograms",Geophysics 53, 1425-1436 – 1988.***

Formulación

En un sistema cartesiano 2D con *X* como eje horizontal y positivo hacia la derecha y *Z* como eje vertical positivo hacia abajo, la ecuación de movimiento de ondas P-SV esta dada por:

$$r\frac{\partial U_t}{\partial t} = \frac{\partial t_{xx}}{\partial x} + \frac{\partial t_{xz}}{\partial z}$$

$$r\frac{\partial W_t}{\partial t} = \frac{\partial t_{zz}}{\partial z} + \frac{\partial t_{xz}}{\partial x}$$

Y las leyes constitutivas en el medio isotrópico:

$$t_{xx} = (l + 2m)\frac{\partial U}{\partial x} + l\frac{\partial W}{\partial z}$$

$$t_{xz} = m\left(\frac{\partial U}{\partial z} + \frac{\partial W}{\partial x}\right)$$

$$t_{zz} = (l + 2m)\frac{\partial W}{\partial x} + l\frac{\partial U}{\partial z}$$

Donde *Ut* y *Wt* son las velocidades de la partícula en x y z respectivamente, *τxx* y *τzz* los tensores de esfuerzos normales, *τzx* el tensor de esfuerzo tangencial, *μ* y *λ* los parámetros de Lame y *ρ* la densidad. La velocidad compresional esta dada por:

$$a = \sqrt{\frac{l + 2m}{r}}$$

Y la velocidad cortante por:

$$b = \sqrt{\frac{m}{r}}$$

Este sistema de ecuaciones diferenciales parciales de primer orden es convertido a un esquema "escalonado" o "desplazado" (en inglés "staggered grid") de diferencias finitas, en el que los parámetros y valores de esfuerzos y velocidades (derivadas del desplazamiento) se calculan en una forma intercalada. Siguiendo el artículo de Levander (1988), para el cálculo de derivadas se aplica una aproximación central en las derivadas espaciales de 4to orden y en la derivada temporal de 2do orden.

Las figuras 17 y 18 muestran la ubicación de las velocidades y tensiones, (U_t, W_t, τ_{xx} , τ_{zz}, τ_{zx}) los 3 parámetros utilizados (μ, λ, ρ) en el esquema escalonado. Las esquinas de los cuadrados son los puntos de la grilla: (m,n), (m+1,n), (m+1,n+1),

(m.n+1). Los componentes de la velocidad son definidos en los tiempos $\Delta t\text{-}1/2$ y $\Delta t+1/2$, y las tensiones en Δt y $\Delta t+1$.

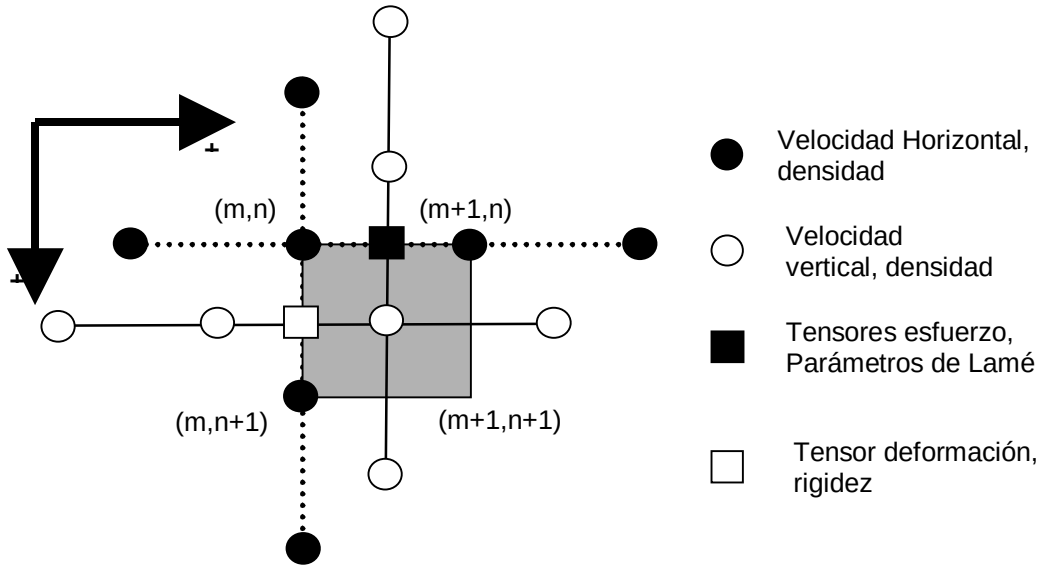


Figura 17 Esquema de la grilla de diferencias finitas y plantilla espacial para la actualización de la velocidad. La plantilla de la velocidad horizontal se muestra como una línea continua, que junto con los nodos de los tensores son utilizados para la actualización. La plantilla de la velocidad vertical se muestra como una línea punteada, que junto con los tensores, son utilizados para la actualización.

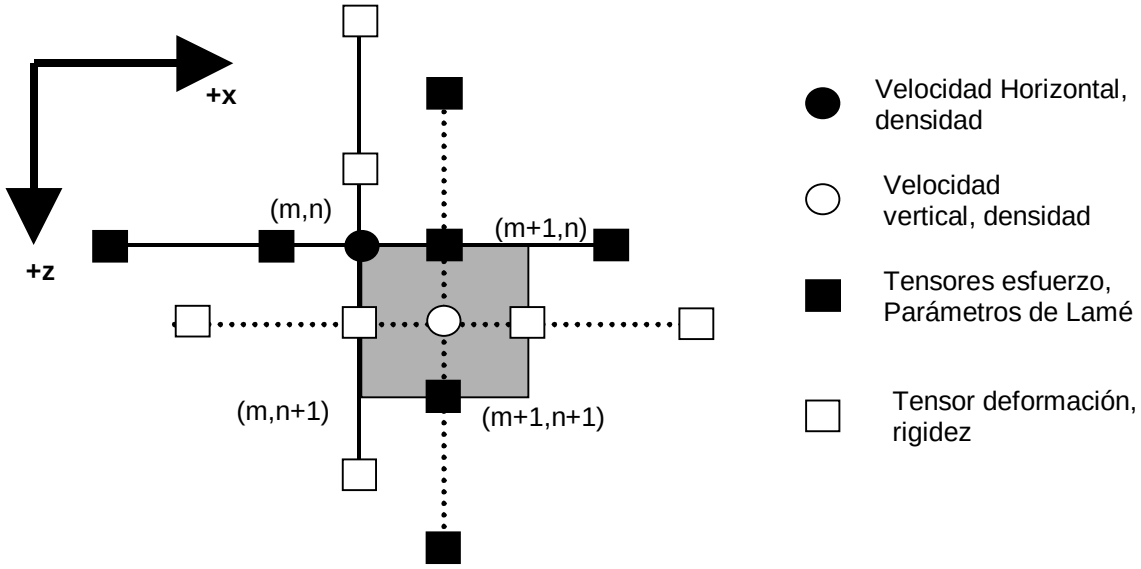


Figura 18 . Esquema de la grilla de diferencias finitas y plantilla espacial para la actualización de los tensores. La plantilla de los tensores de esfuerzo se muestra como una línea continua, que junto con los nodos de los velocidades son utilizados para la actualización. La plantilla del tensor de deformación se muestra como una línea punteada, que junto con las velocidades, son utilizados para la actualización. (Tomado de Levander, 1988)

Condiciones de estabilidad y dispersión

Según lo planteado por Levander (1988), para que el algoritmo funcione correctamente, se deben cumplir las siguientes condiciones de estabilidad y dispersión:

- **ü** Estabilidad

El teorema de equivalencia de Lax

$$\Delta t < \frac{h}{\sqrt{2a\left(c_1 + c_2\right)}}$$

O

$$\Delta t < 0.606h/\alpha$$

Donde c_1 y c_2 son coeficientes de la aproximación de cuarto orden a la primero derivada, (los cuales se calculan con base en método de series de Taylor); h es el mayor delta espacial entre las direcciones x y z y α es la máxima velocidad compresional presente en el modelo.

- **ü** Dispersión:

Para disminuir los efectos de dispersión se requiere que existan a lo máximo 5 celdas de la grilla para representar la menor longitud de onda cortante (shear) presente en el modelo, lo que se puede resumir en la siguiente ecuación:

$$l_{\min} = \frac{b_{\min}}{w_c} > 5h$$

Donde ω_c es la frecuencia central de la ondícula, β_{min} es la menor velocidad cortante (shear) presente en el modelo y h es el delta en el espacio mayor entre Δx y Δz.

Condiciones de Frontera

EcoElas2D puede tomar la frontera superior (la de la superficie) de tres formas diferentes como se describe a continuación:

- **ü** Frontera Absorbente

Trabaja de esta forma cuando el parámetro *fs* es igual a 0 y *higdon* es igual a 1. Cuando está de este modo el programa trata de absorber la ondícula como se describe mas adelante en la sección 1.3.1.

- **ü** Superficie Fija

Se activa cuando el parámetro *fs* es igual a 0 y *higdon* es igual a 0. Cuando está de esta manera, los desplazamientos en toda la frontera superior se hacen igual a 0 con lo cual se provocará una reflexión grande en la superficie.

- **ü** Superficie Libre:

Se establece en este modo cuando el parámetro *fs* es igual a 1 sin importar el valor que tomen los otros parámetros. Cuando trabaja de este modo para todas las celdas de la superficie se hace la siguiente operación:

$$Txz[1][j]=0.0\ ,$$

$$Tzz[1][j]=0.0$$

Donde *Tzz* es el tensor de esfuerzo normal en z y *Txz* es el tensor cortante, *j* es el índice para todas las celdas de la superficie, donde $1 < j < Nx$ *(numero de celdas en x).*

En las fronteras laterales puede tomar las siguientes formas:

- **ü** Se trabaja con las fronteras absorbentes sugeridas por Higdon (1991), o fronteras disipativas.

Fronteras Absorbentes de Higdon

En un modelado numérico de la propagación de ondas en la tierra, es necesario introducir límites computacionales artificiales parar mantener un tamaño manejable. Debido a estos límites computacionales, se produce una reflexión significativa hacia adentro del dominio computacional, pues en la modelo real estas límites no existen. Entonces, se necesitan algún tipo de "barrera" para simular el traspaso de energía hacía afuera del dominio computacional. Estas fronteras son llamadas fronteras absorbentes, pues su función es tratar de "disimular" los efectos que producen este encerramiento y tratar de buscar que el modelo artificial sea lo mas real posible.

Basadas en el artículo de Hidgdon "*Absorbing boundary conditions for elastic waves , Geophysics, Vol 56 pag 231-241"* estas condiciones de frontera se basan en composiciones de operadores diferenciales de primer orden. Estas fronteras son aproximadas con una ecuación de diferencias finitas, que usa los valores de la solución solo a lo largo de líneas de la grilla perpendiculares a la frontera. Esta propiedad facilta su implementación, especialmente cerca de una superficie libre y otras esquinas del dominio computacional.

La característica de fronteras absorbentes se activa cuando el parámetro *higdon* es igual a 1, y utiliza un parámetro adicional denominado *higdon_width* que especifica el número de filas o columnas a las que se le aplicará la frontera absorbente.

Fronteras Disipativas (taper)

Esta característica se activa cuando el parámetro *taper* es igual a 1 y actúa con otros 2 parámetros que son *tw* y *pr.* El parámetro *tw* define el ancho de la zona de frontera en metros, que debería ser aproximadamente 2 longitudes de onda; el punto donde se ubica la fuente debería estar aproximadamente alejado 2

longitudes de onda de esta zona. El parámetro *pr* es el factor de escalamiento que podría dejarse siempre en 25.

Cuando está activa la característica de fronteras disipativas, el programa multiplica por una función exponencial todos los desplazamientos tomando la siguiente forma:

Superficie Libre Activada (fs=1):

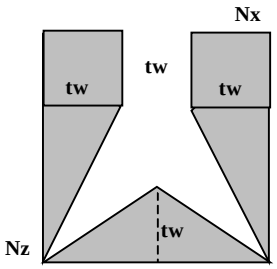


Figura 19. Fronteras Disipativas con superficie libre

Superficie Libre Desactivada (fs=0):

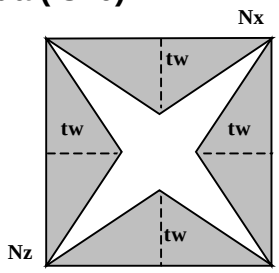


Figura 20 Frontera dispativas sin superficie libre

Donde el área sombreada corresponde a las celdas de la grilla que se multiplicaran por la función exponencial, y *tw* es el numero de celdas que equivalen al ancho de la frontera disipativa dado por el usuario en el parámetro *tw*.

Entradas del programa

El programa como entrada recibe archivos con las Velocidades P, S y las densidades además de los parámetros que se resumen a continuación:

Parámetro	**Descripción**
Nx=100	*cantidad de celdas en X*
Nz=100	*cantidad de celdas en Z*
Dx=0.01	*delta en X*
Dz=0.01	*delta en Z*
Dt=0.01	*delta en el tiempo (seg)*
fcent=30.0	*frecuencia central de la ondicula (Hz)*
Vp=vp.bin	*archivo que contiene las velocidades P*
Vs=vs.bin	*archivo que contiene las velocidades S*
den=density.bin	*archivo que contiene las densidades*
pgeof=pgeof.txt	*archivo que contiene la posición de los geofonos*
par=data.in	*archivo que contiene parámetros a usar*
tmax=1.0	*tiempo de simulación (seg)*
*spx=(nx*dx)/2*	*posición de la fuente en X*
spz=0	*posición de la fuente en Z*
source_type=1	*tipo de fuente* *1 -> punto de fuente dilatacional* *2 -> Verticalmente orientada* *3 ->Horizontalmente orientada*
wavelet_type=1	*tipo de ondicula* *1 -> Segunda derivada de la funcion de Gauss* *2 -> wavelet Kosloff* *3 -> wavelet Ricker* *4 ->wavelet Kuepper*
seismic=1	*determina si graba sismogramas (0=no 1=si)*
step_seismic=0.3	*determina cada cuanto tiempo (seg) graba el sismograma*
snapshots=1	*determina si desea grabar snapshot (0=no 1=si)*
delay_ini=dt	*tiempo de espera para empezar a grabar snapshots*

Parámetro	Descripción
	(seg)
*step_snapshots=5*dt*	*cada cuanto tiempo (seg) graba el un snapshot*
higdon=1	*determina si aplica fronteras absorbentes (0=no 1=si)*
higdon_width=2	*ancho de la frontera absorbente*
fs=1	*determina si aplica superficie libre (0=no 1=si)*
smooth=0	*determina si aplica suavizado a las velocidades y densidades (0=no 1=si)*
taper=0	*determina si aplica fronteras disipativas (0=no 1=si)*
tw=1	*ancho de la frontera disipativa*
pr=0	*porcentaje de disipación*
wg=10e5	*ganancia dada al wavelet*
file_snapx=SnapX.bin	*archivo de salida de los Snapshots en X. Si el usuario utiliza como nombre "null" no se escribirá este archivo*
file_snapz=SnapZ.bin	*archivo de salida de los Snapshots en Z. Si el usuario utiliza como nombre "null" no se escribirá este archivo*
file_snappr=SnapPR.bin	*archivo de salida de los Snapshots Presión. Si el usuario utiliza como nombre "null" no se escribirá este archivo*
file_seismicx=SeismicX.bin	*archivo de salida de la sísmica en X*
file_seismicz=SeismicZ.bin	*archivo de salida de la sísmica en Z*
file_seismicpr=SeismicPR.bin	*archivo de salida de la sísmica Presión*
file_wavelet	*Wavelet.bin*
verbose=1	*determina si imprimir resultados cada dt file_out=data.out archivo de salida del reporte de ejecución*

Tabla 2 Entradas del programa

Salidas del programa

El programa como salida da los siguientes Archivos:

- SnapX.bin SnapShots de los desplazamientos en X.
- SnapZ.bin SnapShots de los desplazamientos en Z.
- SnapPR.bin SnapShots de la presiones calculadas.
- SeismicX.bin Sísmica de los desplazamientos en X.
- SeismicZ.bin Sísmica de los desplazamientos en Z.
- SeismicPR Sísmica de las presiones calculadas.
- Wavelet.bin Ondicula utilizada en el modelamiento.
- Data.out Archivo ascii con el reporte de ejecución.

Con base en la información recopilada en el estudio del algoritmo secuencial, se dio paso al planteamiento del problema y la definición de los requerimientos iniciales.

4.1.2 Requerimientos iniciales:

Una herramienta software para modelar la ecuación de onda completa en 2 dimensiones que además de las características que presenta la herramienta secuencial sea capaz de :

- Ejecutar un problema en menos tiempo del empleado con la aplicación secuencial
- Ejecutar un problema mas eficientemente, a través de la distribución de la carga entre los procesadores.

4.2 DISEÑO

4.2.1 Partición

Las grillas usadas para representar la velocidad de propagación de la onda en X, Z, los tensores de esfuerzo normal y esfuerzo tangenciales, hacen que la descomposición de dominio sea la más adecuada para aplicar en nuestro problema.

Tratando de buscar la mayor concurrencia posible, se ha escogido inicialmente realizar la descomposición más agresiva, que en este caso es definir una tarea por cada punto de la grilla.

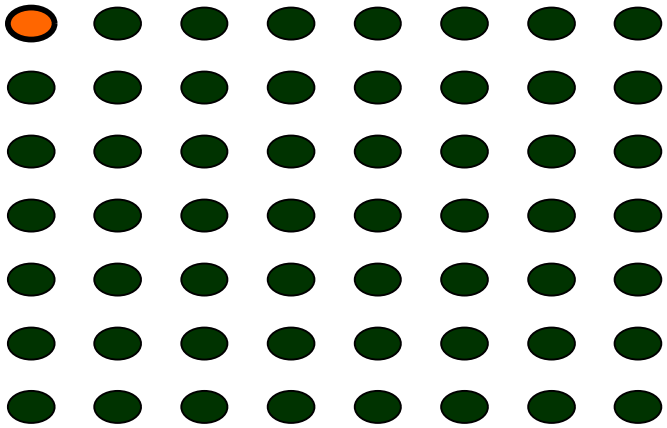


Figura 21 Descomposición del dominio propuesta para Ecoelas2D. Cada tarea es un solo punto de la grilla. La tarea indicada con el punto rojo tiene cinco valores asociados, las cinco variables a actualizar.

Cada tarea mostrada en la figura 21 mantiene actualizado el punto de la grilla, y es responsable el cómputo relacionado con él. La cantidad de tareas que se crean en cada paso de tiempo es Nx x Nz, donde Nx y Nz es la cantidad de puntos en x y z respectivamente.

4.2.2 Comunicación

Para la partición realizada en la parte anterior, se identificaron las siguientes requerimientos de comunicación:

- *Stencil de diferencias finitas:* Debido a que se asumió una descomposición de grano fino. En donde cada tarea encapsula un solo punto de la grilla. Se necesitan 7 puntos de la grilla para realizar la actualización, de los cuales 6 se deben obtener de las tareas vecinas. Los estructura de canales correspondientes se muestran en la figura 22

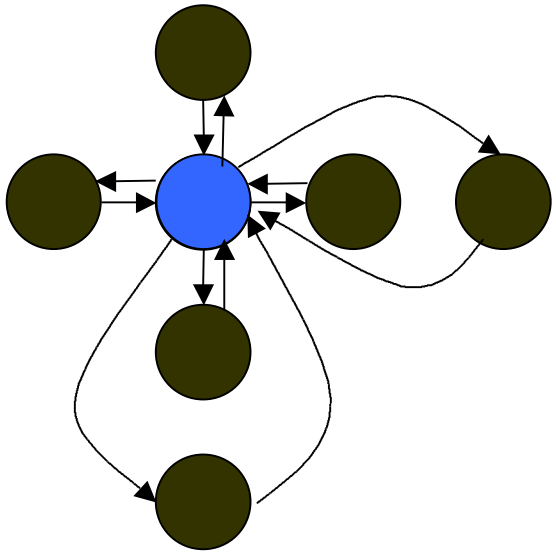


Figura 22 . Tarea y estructura de comunicación para el cómputo de diferencias finitas con siete puntos, asumiendo una tarea por cada punto de la grilla. Solo son mostrados los canales usados por la tarea azul.

- *Comunicación Global*
  - Al inicio de la simulación "todas" las tareas deben tener la información sobre los parámetros de Lamé y la densidad para el punto de la grilla que tiene a cargo. Como estos datos los ingresa el usuario a través de un archivo binario, debe establecerse un canal donde se le entregue los datos a cada una de las tareas.
  - Cada cierto periodo de tiempo, si el usuario indicó la elaboración de sismogramas o snapshots se debe guardar la información de las velocidades en ese paso de tiempo. Para ello se debe crear un

canal, por el cual se enviara la información correspondiente y guardarla en un archivo de salida.

- Se observa que la comunicación entre tareas es bastante grande: Se deben realizar 30 intercambios de datos por punto de la grilla, sin contar la información que debe enviarse para guardar en el archivo.

4.2.3 Agrupación

Con descomposición realizada en la primera etapa se crearían Nx*Nz tareas, por ejemplo para Nx=250 y Nz=250 el número de tareas ascendería a 62500. Esta cantidad de tareas es mucho mayor de la que es necesitada para conservar su flexibilidad. Otras razones identificadas para realizar una agrupación son las siguientes:

- Una agrupación de tareas como las mostradas en la figura puede reducir los requerimientos de comunicación asociados de 12 intercambios por paso de tiempo a 6 por tarea.
- El lenguaje en el cual fue desarrollada la versión secuencial de EcoElas2D fue C y la forma de representar la grilla es una matriz. En este lenguaje, las matrices son almacenadas en memoria por filas.
- El manejo de los 3 tipos de fronteras de simplificaría considerablemente, si disminuyera la dimensión del dominio.

Este análisis hace pensar, que para “acomodar” nuestro algoritmo paralelo se debe utilizar un modelo de descomposición de la grilla en una dimensión (horizontal), Los requerimientos de comunicación se reducen a la mitad. El número de filas que tendrá cada procesador, será determinado en la siguiente fase.

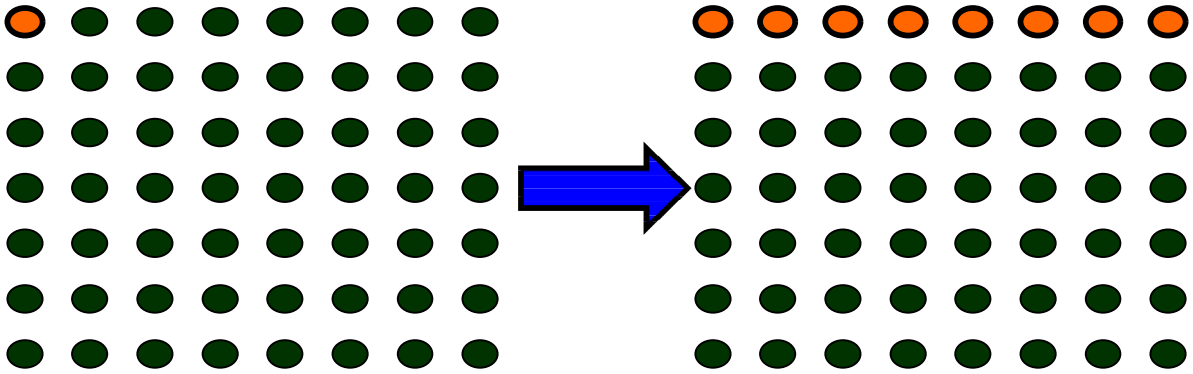


Figura 23 En la figura se observa el aumento en la dimensión para la grilla. En la parte izquierda para la partición que realizada en la primera etapa, y en la parte derecha se muestra la nueva partición en forma horizontal

4.2.4 Asignación

Para evitar desbalanceo de carga, la ha realizado la estrategia de asignación de la figura. Dependiendo del numero de procesadores que se escojan para realizar el computo; el algoritmo repartirá el número de tareas correspondientes a cada procesador. Para evitar comunicación entre procesadores, se busca que las tareas asignadas a cada procesador sean "vecinas" y solo se produzca comunicación en la frontera de arriba y abajo.

El manejo de la condiciones de frontera las asumirá cada procesador según le corresponda.

4.3 IMPLEMENTACIÓN

Teniendo en cuenta el diseño realizado, se procedió a la etapa de codificación y depuración del algoritmo y cada una de las librerías que componen la aplicación.

4.3.1 Lenguaje de Programación

El lenguaje de programación empleado fue gcc 3.2.5. Gcc es el estándar diseñado para el sistema operativo Linux, siendo este que brinda el ambiente

necesario para el desarrollo de la computación paralela. La razón por la cual se escogió este lenguaje, es porque el algoritmo secuencial esta implementado en gcc y la librería que permite la comunicación el paso de mensajes entre los procesadores LAM-MPI2 tiene una versión para gcc.

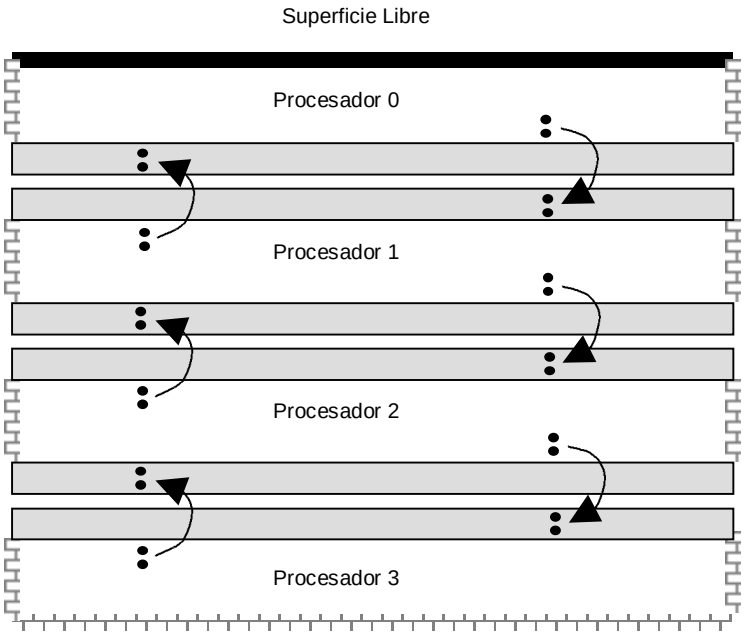


Figura 24 Descomposición de la grilla global, en los dominios locales que le corresponden a cada procesador. El intercambio entre los procesadores vecinos se debe realizar en cada paso de tiempo.

Respecto a la escogencia de MPI como la estándar para la comunicación entre los procesadores, se realizó en base al tipo de máquina paralela donde se ejecutaría la aplicación, el cluster del ICP posee una memoria distribuida. La librería LAM-MPI es la que se encuentra instalada en el cluster.

4.3.1 Formato datos de entrada

Se conservó el formato de los datos de entrada de la versión secuencial, agregando un nuevo dato llamado np que le indicará el número de procesadores que se desean utilizar para ejecutar la aplicación.

- **ü** Versión secuencial:

ecoelas2d nx= nz= dx= dz= dt= fcent= vp= vs= den= pgeof= [Parámetros opcionales].

Ó

ecoelas2d par=[Archivo ASCII con todos los parámetros requeridos y opcionales]

- **ü** Versión paralela

ecoelas2dp nx= nz= dx= dz= dt= fcent= vp= vs= den= pgeof= np= [Parámetros opcionales]

ó

ecoelas2dp np= par=[Archivo ASCII con todos los parámetros requeridos y opcionales]

En esta versión solo se manejara llamado por consola, para futuras versiones se sugiere que se diseñe una interfaz gráfica para mayor comodidad del usuario.

4.3.2 Implementación del algoritmo

Ecoelas2d esta divido en varios librerías en donde se encuentran diferentes funcionalidades, que son usadas en el programa principal como se explica en la siguiente tabla:

PROGRAMA /LIBRERIA	DESCRIPCION
Ecoelas2d	programa principal (Foco de la la paralelización)
Par	Se encuentran las funciones de asignación de memoria dinámica
Taper	Se encuentran las funciones que manejan las fronteras disipativas
Higdon	Se encuentran las funciones que manejan Fronteras absorbentes
Wavelet	Se encuentran las funciones que manejan el tipo de ondícula
Derivatives	Se encuentran las funciones que manejan las derivación en el espacio de 4to orden

Tabla 3 Librerías y programas que componen a Ecoelas

A continuación se describirán los cambios que se efectuaron a cada uno de los algoritmos.

1. Ecoelas2d

Para realizar el control del algoritmo, se hizo necesario nombrar un “maestro” que es el computador que tendrá la iteración con el usuario, y asignará a los demás “trabajadores” las tareas que deben realizar. Al final cada uno de los trabajadores devolverá al maestro la tarea realizada.

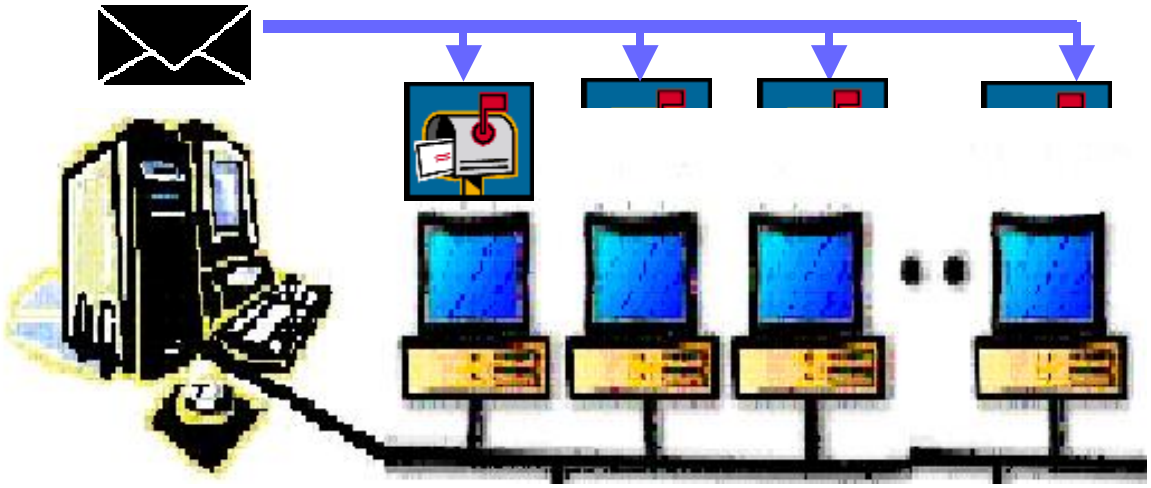


Figura 25 Distribución de las tareas por parte del computador maestro a cada uno de los trabajadores.

Las modificaciones que se realizaron en el programa principal se han divido en 2 partes:

- *Modificaciones a nivel global:* Estas modificaciones se refieren a las que se tienen que hacer a lo largo del programa, debido al cambio de paradigma. Por ejemplo, la lectura de los parámetros iniciales que debe dar el usuario al ejecutar la aplicación son manejados solo por el maestro, esto no debe importarle a los nodos trabajadores, ellos solo esperan que el maestro, les "entregue" a información que tiene que procesar.

- *Modificaciones a nivel "administrativo":* Estas modificaciones se refieren a las que se tienen que hacer debido a la intercambio de información entre el maestro y los trabajadores. Estas modificaciones se traducen en la creación de 12 nuevas variables, 10 funciones y aproximadamente 300 líneas de código (Ver Tabla 4)

FUNCIONES	USO
Envio_parametros	La ejecuta el maestro, con esta función se envía los primeros parámetros a los trabajadores
Envio_matrices	La ejecuta el maestro, con esta función se envía los pedazos de la matriz que se deben calcular a los trabajadores.
Envio_coeficientes	La ejecuta el maestro, con esta función se envía los coeficientes de las fronteras de Higdon, si es pertinente.
Recibo_parametros	La ejecuta los trabajadores, con esta función reciben los primeros parámetros.
Envio_matrices	La ejecuta los trabajadores, con esta función se reciben los pedazos de la matriz que se deben calcular.
Envio_coeficientes	La ejecuta los trabajadores, con esta función se recibe los coeficientes de las fronteras de Higdon.
Intercambio	Con esta función intercambio las filas entre mis vecinos de abajo y arriba
Init_Line	Con esta función se hace la división del dominio entre todos los nodos que tenga disponibles

Tabla 4 Funciones programa principal

VARIABLES	USO
posiciones[i]	Vector tipo entero donde se guarda la fila inicial y la fila final de cada uno de los procesadores
my_id	Variable tipo entera donde se guarda el identificador de cada procesador. El computador principal tendrá como identificador el cero (0).
id_fuente	Variable tipo entera en donde se guardar el id del procesador que posee el punto donde se encuentra la fuente
arriba	Variable tipo entera donde se guarda el id del procesador con el que tengo que intercambiar las filas arriba.

VARIABLES	USO
abajo	Variable tipo entera donde se guarda el id del procesador con el que tengo que intercambiar las filas abajo.
cuantas	Variable tipo entera donde se guarda la cantidad de filas que fueron asignadas a cada procesador.

Tabla 5 Variables creadas para la paralelización

2. Par

A esta librería no hubo necesidad de realizar cambios, debido a que cada una las funciones especifica la cantidad de memoria a reservar.

3. Taper

A esta librería se le realizaron cambios a las 2 de las 3 funciones que posea, ya que esta librería maneja las fronteras disipativas, y estas debe aplicarlas cada procesador según sea el caso. En la siguiente tabla se muestra las funciones con los cambios que se realizaron a cada una de ellas.

ANTES		AHORA	
InitTaper	Calcula los coeficientes de disipación para cada una de las filas void InitTaper(float *Coeff, int Count, float MinCoeff)	InitTaper	Tiene la misma estructura, aunque ahora solo la ejecuta

ANTES		AHORA	
Taper	Aplica el coeficiente de disipación a los desplazamientos *Void TaperX(float **Ux, float **Uz, float *Coeff, int nx, int nz, int nCoeff)*	Taper	Realiza la misma función, solo que ahora tiene 3 argumentos más: *Void*
Taper4d	Aplica el coeficiente de disipación a los desplazamientos, en el caso que no exista una superficie libre. *void Taper42d(float **Ux, float **Uz, float *Coeff, int nx, int nz, int nCoeff)*	Taper4d	Realiza la misma función, solo que ahora tiene 3 argumentos más:

Tabla 6 Cambios Realizados librería Taper

4. Higdon

A esta librería se le realizaron cambios a las 2 de las 3 funciones que posea, ya que esta librería maneja las fronteras absorbentes, y estas debe aplicarlas cada procesador según sea el caso. En la siguiente tabla se muestra las funciones con los cambios que se realizaron a cada una de ellas.

ANTES		AHORA	
Absorbing	Calcula los parámetros de Higdon para cada una de las fronteras *void Absorbing(float **Lambda2Mu, float **Mu, float **Ro, float dx, float dz, float *tr1, float *tr2, float *tb, float *br1, float *br2, float *bb, float *lr1, float *lr2, float *lb, float *rr1, float *rr2, float *rb, int nx, int nz, float dt, int fs)*	Absorbing	Tiene la misma estructura, aunque ahora solo la ejecuta el computador maestro
HigdonX	Aplica Higdon a los desplazamientos en X *void HigdonX(float **Ux, float **Uxo, float **Uxom, float *tr1, float *tr2, float *tb, float *br1, float *br2, float *bb, float *lr1, float *lr2, float *lb, float *rr1, float *rr2, float *rb, int nx, int nz, int fs, int width)*	HigdonX	Realiza la misma función, solo que ahora tiene 3 argumentos más: *void HigdonX(float **Ux, float **Uxo, float **Uxom, float *tr1, float *tr2, float *tb, float *br1, float *br2, float *bb, float *lr1, float *lr2, float *lb, float *rr1, float *rr2, float *rb, int nx, int nz, int fs, int width, int *posiciones, int my_id, int tamano)*

HigdonZ	Aplica Higdon a los desplazamientos en Z *void HigdonZ(float **Uz, float **Uzo, float **Uzom, float *tr1, float *tr2, float *tb, float *br1, float *br2, float *bb, float *lr1, float *lr2, float *lb, float *rr1, float *rr2, float *rb, int nx, int nz, int fs, int width)*	HigdonZ	Realiza la misma función, solo que ahora tiene 3 argumentos más: *void HigdonZ(float **Uz, float **Uzo, float **Uzom, float *tr1, float *tr2, float *tb, float *br1, float *br2, float *bb, float *lr1, float *lr2, float *lb, float *rr1, float *rr2, float *rb, int nx, int nz, int fs, int width, int *posiciones, int my_id, int tamano*

Tabla 7 Cambios Realizados A librería Higdon

5. Wavelet

A esta librería no se le realzaron modificaciones, ya que las funciones que se encuentran alli solo las ejecuta el computador maestro, de forma secuencial

6. Derivatives

A esta librería se le realizaron cambios a las 4 de las 8 funciones que posea, debido a que se realizó una asignación de tareas por filas no hubo la necesidad de modificar las funciones que calculan las deriviadas en X. En la siguiente tabla se muestra las funciones con los cambios que se realizaron a cada una de ellas

ANTES		AHORA	
Dxm4	Halla una aproximación a la derivada en X de la matriz IN Void dxm4(float **IN, float **OUT, int nx, int nz, float dx, int xs, int xf, int zs,int zf)	Dxm4	Igual, la ejecuta todos los nodos
Dxm4m	Halla una aproximación a la derivada en X de la matriz X y le suma los valores e la matriz OUT y lo multiplica por el valor de la matriz M Void dxm4m(float **IN, float **OUT, float **M, int nx, int nz , float dx, int xs, int xf, int zx, int zf)	Dxm4m	Igual, la ejecuta todos los nodos
Dxp4	Halla una aproximación a la derivada en X de la matriz IN void dxp4(float **IN,int nx, int nz, float dx, int xs, int xf, int zs,int zf)	Dxp4	Igual, la ejecuta todos los nodos

ANTES		AHORA	
Dxp4m	Halla una aproximación a la derivada en X de la matriz X y le suma los valores e la matriz OUT y lo multiplica por el valor de la matriz M Void dxp4m (float **IN, float **OUT, float **M, int nx, int nz , float dx, int xs, int xf, int zx, int zf)	Dxp4m	Igual, la ejecuta todos los nodos
Dzm4	Halla una aproximación a la derivada en Z de la matriz IN Void dzm4(float **IN, float **OUT, int nx, int nz, float dz, int xs, int xf, int zs, int zf, int fs)	Dzm4	Realiza la misma función, solo que ahora tiene 3 argumentos más: Void dzm4(float **IN, float **OUT, int nx, int nz, float dz, int xs, int xf, int zs, int zf, int fs, int *posiciones, int my_id, int tamano)
Dzp4	Halla una aproximación a la derivada en Z de la matriz IN Void dzp4(float **IN, float **OUT, int nx, int nz, float dz, int xs, int xf, int zs, int zf, int fs)	Dzp4	Realiza la misma función, solo que ahora tiene 3 argumentos más: Void dzp4(float **IN, float **OUT, int nx, int nz, float dz, int xs, int xf, int zs, int zf, int fs, int +posiciones, int my_id, int tamano)

Tabla 8 Cambios Realizados librería Derivatives

5. PRUEBAS

El objetivo de la fase de pruebas, es realizar alguna medida de rendimiento que nos permita comparar los tiempos de ejecución del algoritmo secuencial con los tiempos de ejecución del algoritmo paralelo. Para llevar a cabo esta fase de la investigación se diseño el siguiente plan de pruebas:

5.1 LEY DE AMDAHL

La aceleración de un algoritmo paralelo esta definida como:

$$S = \frac{\textit{Tiempo de ejecución secuencial}}{\textit{Tiempo de ejecución paralela}}$$

Esta medida experimental, se compara con los resultados obtenidos aplicando la ley de Amdahl's[2].

La ley de Amdahl's postula que la aceleración máxima que puede alcanzar un algoritmo paralelo es definida por la fracción del código (P) la que es paralelizada:

$$S = \frac{1}{1 - f}$$

- Si nada del código puede ser paralelizado, f = 0 y la aceleración = 1 (aceleración nula). Si todo el código es paralelizado, f = 1 y la aceleración es infinita (en teoría). Si 50% del código puede ser paralelizado, la aceleración máxima = 2, lo cual significa que el código se ejecutará el doble de rápido (medida ideal).

- Introduciendo el número de procesadores ejecutando la fracción paralela de trabajo, la relación puede ser modelada por:

[2] The validity of the single processor approach to achieving large scale computing capabilities. En: AFIPS Conference Proceedings, Spring Joint Computing Conference

$$S = \frac{p}{f + \frac{1 - f}{p}}$$

En donde S es la aceleración, f la fracción del código ejecutado en paralelo, f es la fracción del código que corre secuencialmente, p el número de procesadores utilizados para la ejecución.

5.2 COMUNICACIÓN Y REQUERIMIENTOS DE MEMORIA

La comunicación juega un papel importante en cualquier aplicación dependiente de la red. Para este caso, la comunicación es un factor crítico, pues en cada paso de tiempo se deben intercambiar las filas entre los procesadores vecinos.

Por esto en esta etapa se hace una análisis de la cantidad de información a ser intercambiada en cada paso de tiempo, así como los requerimientos de memoria para cada procesador.

5.3 EJECUCION DEL ALGORITMO

Las pruebas se realizaron, en un cluster de 5 PCs con sistema operativo Linux Red Hat 9.0 conectados a través de una red Ethernet de 100 Mb,/s con una memoria RAM de 256 Mb y un procesador de 300 MHz.

6. RESULTADOS

6.1 DEMO 1

Las características del modelo son:

- Grilla de 250*250 en la dirección Nz Nx respectivamente.
- El espaciado del la grilla es 4 m.
- Se aplicó la condición de superficie libre y frontera Absorbente de Higdon.
- El tiempo de la simulación fue 2 segundos
- Paso de tiempo de 0.001
- Posición de la fuente x=500 m z=0m
- Posición geófonos Están situados en una línea horizontal a lo largo de la superficie, espaciados cada 5 metros en la dirección del eje X.
- Tipo de ondícula Ricker
- Paso de tiempo para guardar sismogramas 0.002 s
- Paso de tiempo para guardar snapshots 0.005s

La estructura sobre el modelo se puede ver en la figura

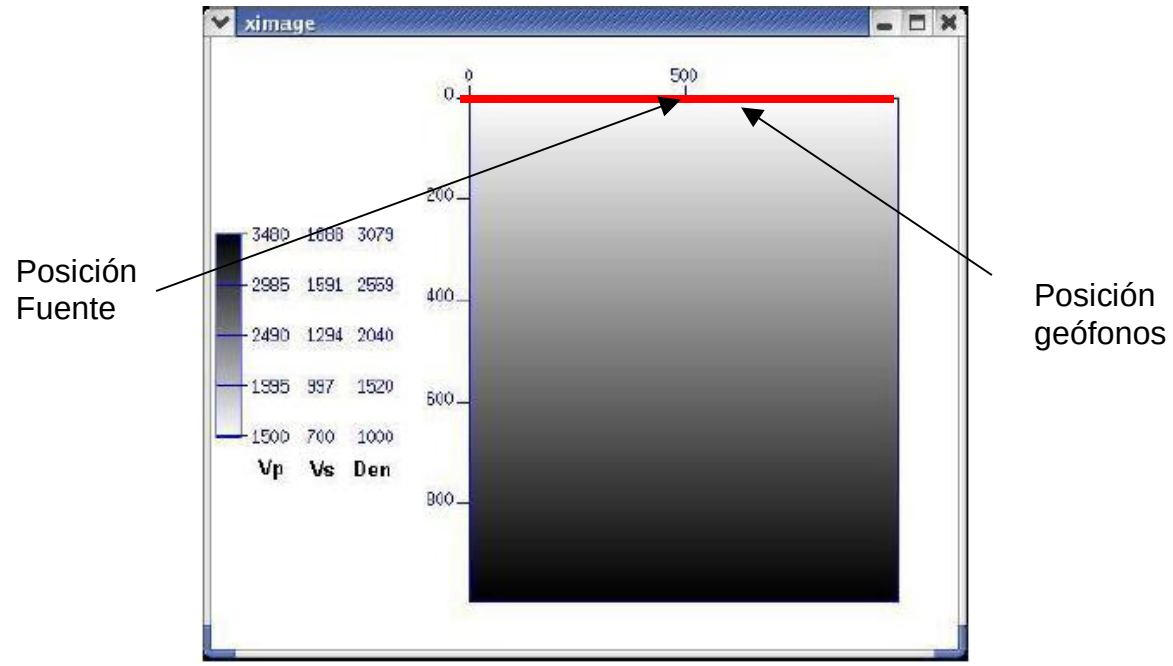


Figura 26 Estructura del modelo, velocidades p, s y la densidad. La aplicación trabaja con archivos de entrada y salida con el formato de SU que hace muy fácil la visualización de los resultados.

6.1.1 Aceleración

Los resultados obtenidos corriendo la aplicación con 2, 3,4 y 5 nodos se resumen en la siguiente gráfica. La línea de color fucsia representa los valores de la aceleración calculada por ley de Amdahl's, con un porcentaje de código paralelo del 25% de, mientras la línea azul muestra la aceleración experimental.

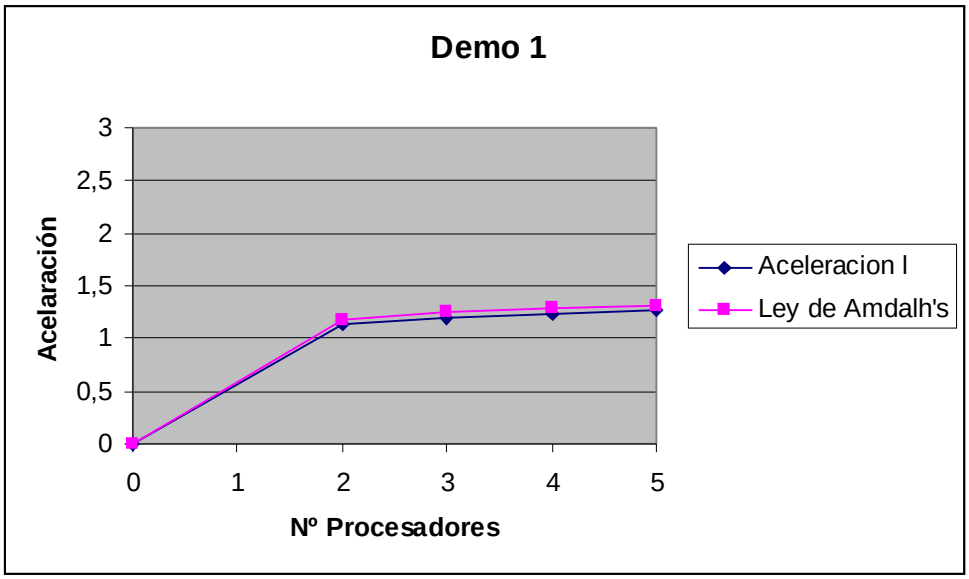


Figura 27 La aceleración para el demo1. Observe que a medida que se agregan procesadores la aceleración va creciendo.

La aceleración para el demo1 va en aumento, pero no es el esperado. Esto se debe a que el tiempo disminuyó (Como lo muestra en la figura 27) pero no de forma dramática. Ahora, el tiempo total también se ve afectado por la comunicación en la red, el cluster donde se realizaron las pruebas no esta conectado con una red de alta velocidad.

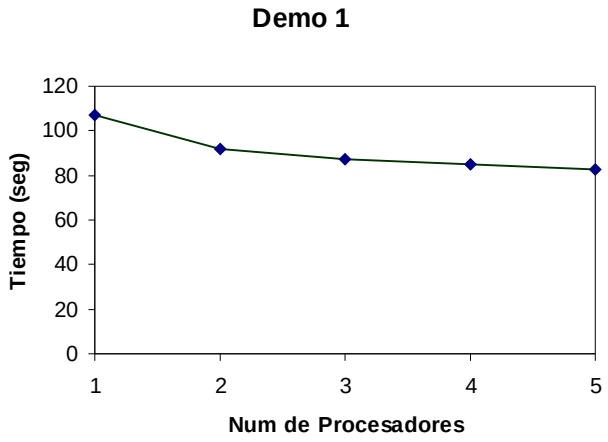


Figura 28 Tiempos totales para el demo 1

Se puede observar de la gráfica que la reducción de tiempo para este modelo en especial no es muy significante, esto debido que la simulación es muy pequeña.

Este ejemplo nos hace pensar que los modelos que se deben ejecutar en una aplicación paralela deben ser suficientemente grandes, para que se note una reducción considerable en el tiempo.

6.2 DEMO 2

Las características del modelo son:

- Grilla de 500*500 en la dirección Nz Nx respectivamente.
- El espaciado del la grilla es 2 m.
- Se aplicó la condición de superficie libre y frontera Disipativas.
- El tiempo de la simulación fue 2 segundos
- Paso de tiempo de 0.0002
- Posición de la fuente x=500 m z=0m
- Posición geófonos Están situados en una línea horizontal a lo largo de la superficie, espaciados cada 20 metros en la dirección del eje X.
- Tipo de ondícula Ricker

- ü Paso de tiempo para guardar sismogramas 0.02 s
- ü Paso de tiempo para guardar snapshots 0.1s

La estructura sobre el modelo se puede ver en la figura

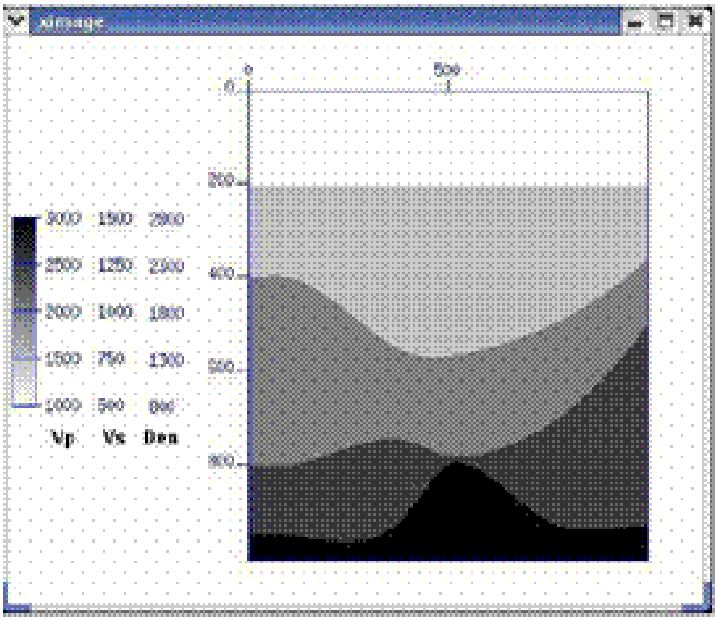


Figura 29 La estructura del modelo, velocidades p, s y densidad

6.2.1 Aceleración

Los resultados obtenidos corriendo la aplicación con 2, 3, 4 y 5 nodos se resumen en la siguiente gráfica. La línea de color fucsia representa los valores de la aceleración calculada por ley de Amdahl's, con un porcentaje de código paralelo del 60% , mientras que la línea azul muestra la aceleración experimental.

La diferencia tan grande entre la fracción código paralelizable en el demo1 y en el demo2 se debe a que se realizó una descomposición por dominio, y esto hace que la cantidad de código que se puede paralelizar dependa del tamaño de la grilla con la cual se va a trabajar. Por eso se hace necesario que los modelos que se deseen simular con la aplicación paralela tengan el suficiente grado de

complejidad, para asi poder aprovechar las ventajas brindadas por la paralelización.

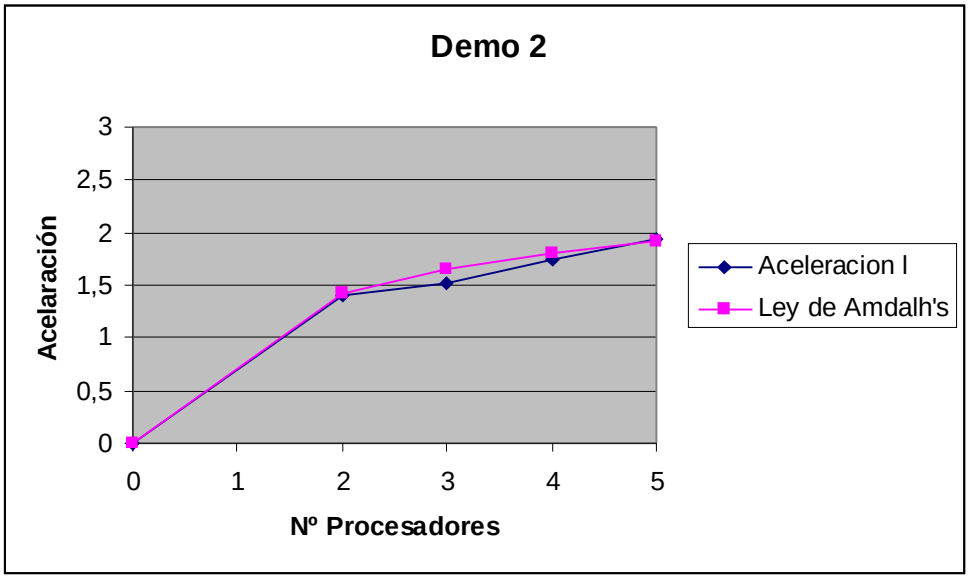


Figura 30. La aceleración para el demo2 El porcentaje de código paralelo para este modelo esta alrededor del 60%..

En la visualización de los tiempos de ejecución se puede ver que se reducen casi a la mitad. Debido que la grilla es mucho más grande y el tiempo de simulación es 100 veces más grande que el modelo anterior.

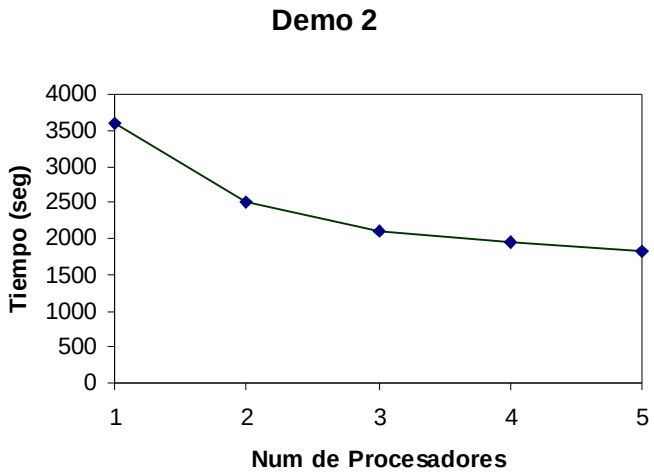


Figura 31 Tiempos de ejecución para el demo 2

6.3 COMUNICACIÓN Y REQUERIMIENTOS DE MEMORIA

Se realizó un análisis de la comunicación en cada paso de tiempo, para determinar que la cantidad de información que se debe intercambiar en cada paso de tiempo, como se muestra en la siguiente figura.

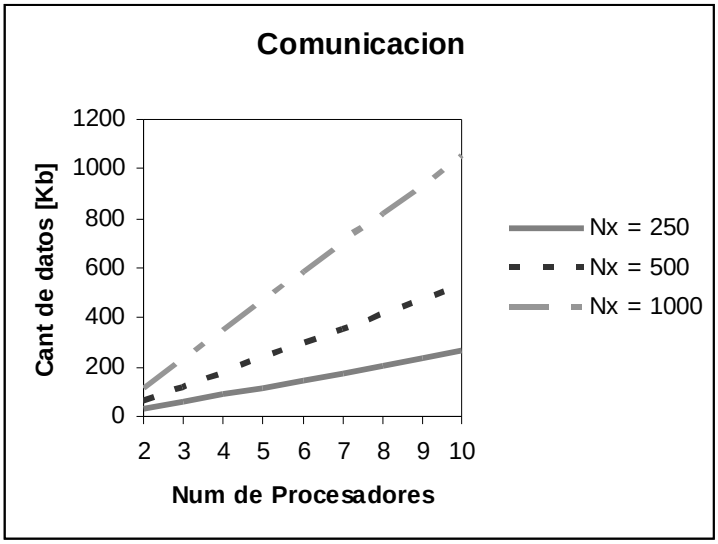


Figura 32 . Cantidad de datos que se intercambian en cada paso de tiempo por los procesadores. Este valor depende de Nx . En la grafica se muestra para valores de Nx=250 Nx=500 y Nx=1000

Por ejemplo, se puede observar que para un Nx=1000 la cantidad de información a intercambiar en cada paso de tiempo asciende a 1Mb por cada iteración.

Para evitar que la cantidad de información haga que se pierda tiempo de procesamiento, se adoptó la estrategia de realizar la menor cantidad de envíos, pero con la misma cantidad de información.

Por ejemplo en Ecoelas las 5 variables a actualizar son:

- Txx
- Tzz
- Txz
- Ux

- **ü** Uz

Esto implicaría que se realizarían 5 envios por cada paso de tiempo. Para aumentar la eficiencia del algoritmo, se realizo un “empaquetamiento” de los datos a enviar y se realiza un solo envio en cada paso de tiempo.

La cantidad de memoria utilizada por cada procesador disminuye considerablemente, debido a que cada uno reserva solo la memoria que necesita, haciendo que se pueda aumentar los tamaños de la grilla. Los requerimientos de memoria se pueden calcular a través de la siguiente fórmula:

$$\text{Memoria}_{\text{procesador}} = \frac{92 * Nx * Nz}{\text{Memoria}/^3} \text{ bytes}$$

Esta ecuación puede ayudar a determinar cuantos procesadores son necesarios para realizar una simulación de Nx *Nz puntos de grilla.

[3] Esta ecuación se deduce de la cantidad de variables que utiliza la aplicación

7. CONCLUSIONES

1. Los ejemplos y las experiencias anteriores demuestran que el esquema de diferencias finitas escalonado es eficiente, y se puede aplicar fácilmente esta técnica para modelos 3D, viscoelásticos, o con algún grado de anisotropía. y heterogeneidad.

2. El técnica de diseño de algoritmos paralelos planteado por Ian Foster, es una herramienta que permite una mayor comprensión del problema, haciendo más fácil la tarea del diseño del algoritmo paralelo.

3. A través de este proyecto de investigación se muestra la efectividad de la implementación del código paralelo a través de la interfaz de paso de mensajes (MPI) , haciendo una división del dominio a lo largo del eje Z. Realizando la descomposición de esta manera, evito mayor comunicación entre los procesadores haciendo mas eficiente el algoritmo.

4. Las pruebas realizadas muestran que para una mayor eficiencia de EcoElas2D se debe ejecutar en redes de alto desempeño, o bien a través de maquinas creadas para estos propósitos.

5. Para aprovechar al máximo los beneficios del Ecoelas2D versión paralela, se debe tener un tamaño de grilla mínimo de 500*500 y un tiempo de simulación de más de 1000 iteraciones.

6. La utilización de Clusters Linux, es una importante opción a tener en cuenta, ya que tanto el software como el hardware tienen un costo relativamente bajo, frente a las otras opciones del mercado (mainframes, supercomputadoras, etc)

7. Este trabajo de investigación es la primera versión paralela, se espera que se realice la segunda versión implementando algunas mejoras que se plantean en las recomendaciones.

8. RECOMENDACIONES

Este trabajo es un primer acercamiento al desarrollo de aplicaciones paralelas para geofísica. Debido a la complejidad del problema, hubo la necesidad de delimitar el alcance de este proyecto, por lo cual se sugiere que sean tenidas en cuenta las siguientes recomendaciones:

1. Realizar un modelo geológico real que permita validar los resultados obtenidos a través del desarrollo de este proyecto de investigación

2. Estudiar la factibilidad de aplicar la técnica de paralelización de diferencias finitas para problemas 3D tanto en medio visco-elásticos, acústicos o con algún grado de anisotropía o heterogeneidad.

3. . Realizar las siguientes mejoras a Ecoelas2D versión paralela
   - ü Sea creada una interfaz grafica, para comodidad para el usuario.
   - ü Se implementen las características de Ecoelas2D 2.0: superficie rugosa, y grilla variable.
   - ü Se implemente un sistema de archivos tipo NFS para guardar la información de sismogramas y snapshot en cada procesador

BIBLIOGRAFIA

Libros

1. **BRICEÑO Y CUELLLAR, Ensayos de Reflexion y Refraxión Sísmica. Curso de Interconexion Eléctrica S.A /ISA. Febrero 8 al 18 1990.**

2. **FOSTER, Ian. Designing and Building Parallel Programs, Addison-Wesley. 1995.**

3. **GETTYS E, Física Clasica y Moderna, McGrawHill 1991.**

4. **HOFFMAN, Joe. Numeral Methods for Engineers and Scientists. Ed Marcel Dekker 2001**

5. **KARNIADAKIS, George y KIRBY, Robert. Parallel Scientific Computing in C++ and MPI. Ed. Cambridge University. 2001**

6. **REYNOLDS, JOHN M. An Introduction To Applied An Environmental Geophysics. Ed John Wiley 1998.**

Tesis de Grado

1. **BARRIOS, Carlos y CASALLAS, Juan Carlos. Desarrollo de aplicaciones para el procesamiento paralelo en una red de PCS. Universidad Industrial de Santander. 2002.**

Artículos

1. **BOHLEN, Tomas, Parallel 3-D viscoelastic finite difference seismic modeling, Computers and Geosciencies Vol 28,2002**

2. **Higdon, R. L.Absorbing boundary conditions for elastic waves. Geophysics, Vol 56 No 2 Febrero 1991.**

3. **Levander Alan R, Fourth-orden finite-diference P_SV seismograms. Geophysics, 53, 1425-1436.**

4. **Virieux, Jean, 1986, P-SV wave propagation in heterogeneous media: "Velocity-stress finite-difference method", Geophysics 51, 889-901**

Manuales

1. **Córdoba, Félix. EcoElas2D 1.0: Manual del Usuario. Octubre 2003.**

2. **Córdoba, Félix. EcoElas2D 1.0: Manual del Administrador. Octubre 2003.**

Páginas de Internet

1. Página oficial LAM-MPI www.lam-mpi.org

ANEXOS

ANEXO A

ARTÍCULO PRESENTACIÓN ORAL II CONGRESO LATINOAMERICANO DE SISMOLOGIA

Computación avanzada aplicada al modelamiento de ondas elásticas

Diana C Hortúa*, Saúl E. Guevara**, Felix Córdoba*** Henry Lamos Díaz****
* UIS, Escuela de Ingeniería de Sistemas
**** UIS, Escuela de Matemáticas
** ECOPETROL-ICP
*** Numérica Ltda..

Resumen
El poder de cómputo ha avanzado a un estado donde podemos comenzar a realizar simulaciones cada vez más reales. Así, para modelar la propagación de ondas sísmicas en medios complejos se creó el software EcoElas2D, que utiliza el método de diferencias finitas para simular la propagación de ondas elásticas en dos dimensiones (2D). Sin embargo, en plataformas seriales los cálculos todavía se limitan a los tamaños pequeños en caso que el modelo exija una grilla muy fina, y/o un pequeño intervalo de tiempo. Este es un proceso típico de alta exigencia en CPU. La paralelización, una opción que permite incrementar el desempeño de la computación, se aplicó a un prototipo de este software. Para hacer uso del poder de cómputo de los clusters, Ecoelas2D distribuye el trabajo sobre cada uno de los procesadores que lo componen. Usando el Interfaz de paso de mensajes (MPI) para la comunicación entre los procesadores, los tiempos de ejecución se reducen, así como los tamaños del modelo, aún para grillas pequeñas, se puede aumentar considerablemente. Una comparación de desempeño permite vislumbrar las ventajas de esta técnica.
Adicionalmente se implementó una versión más avanzada de este software que permite tener en cuenta topografía abrupta y hacer variaciones en el tamaño de la grilla a través del volumen. Esto permite optimizar el tamaño de la grilla y por lo tanto reducir el costo computacional. Un futuro desarrollo de esta última versión de EcoElas2D es también proceder a una paralelización.

1. Introducción

A partir de la teoría de propagación de las ondas es posible simular el comportamiento de una perturbación sísmica a través de un medio de características físicas conocidas. Esta es la técnica conocida como modelamiento sísmico. El modelamiento es una herramienta de gran utilidad en el estudio de la sismología aplicada. En particular en el caso de la exploración de recursos naturales de la tierra por medio de sísmica, permite hacer pruebas de hipótesis acerca de las características geológicas y conocer los efectos de estas características.

Los modelos teóricos pueden tener una gran variedad y diferentes grados de exactitud y esto se refleja en los diferentes algoritmos existentes en el modelamiento sísmico. Uno de ellos, el trazamiento de rayos, se basa en un modelo muy simplificado, pero que tiene grandes ventajas desde el punto de vista computacional. El método de Diferencias Finitas es una opción mucho más exacta, ya que se basa en la ecuación de Onda completa, pero tiene limitaciones en su aplicación práctica debido a que requiere grandes recursos computacionales.

Teniendo en cuenta su exactitud, en el ICP de ECOPETROL SA se implementó un algoritmo de diferencias finitas 2D para fines experimentales. Es un algoritmo elástico 2D, basado en el artículo de Levander (1988). El principal objetivo es utilizarlo para evaluar la respuesta sísmica en las zonas complejas de Colombia, en donde existe la necesidad de mejorar la imagen que se obtiene con el método sísmico. Estas zonas se caracterizan por la compleja estructura geológica en el subsuelo y la topografía rugosa en superficie. Este algoritmo es probablemente uno de los más indicado para este tipo de situación.

Con el objetivo de aumentar su eficiencia computacional, se planteó la necesidad y utilidad de la paralelización, cuyo resultado se presenta en este trabajo. En este artículo se presenta primero la teoría básica del algoritmo y más adelante su proceso de paralelización. Finalmente se habla de otros avances y posibles desarrollos futuros, y se presentan unas conclusiones y recomendaciones.

2. Teoría

1.1 Ecuaciones de onda

Las ecuaciones que describen la propagación de ondas elásticas en un medio isotrópico, implementadas en el algoritmo objeto de este trabajo, son:

a) Ecuaciones de movimiento:

$$r\frac{\partial U_t}{\partial t} = \frac{\partial t_{xx}}{\partial x} + \frac{\partial t_{xz}}{\partial z}$$

$$r\frac{\partial W_t}{\partial t} = \frac{\partial t_{zz}}{\partial z} + \frac{\partial t_{xz}}{\partial x} \qquad (1)$$

b) leyes constitutivas en el medio isotrópico:

$$t_{xx} = (l+2m)\frac{\partial U}{\partial x} + l\frac{\partial W}{\partial z}$$

$$t_{xz} = m\left(\frac{\partial U}{\partial z} + \frac{\partial W}{\partial x}\right) \qquad (2)$$

$$t_{zz} = (l+2m)\frac{\partial W}{\partial x} + l\frac{\partial U}{\partial z}$$

Donde *Ut* y *Wt* son las velocidades de la partícula en x y z respectivamente, *τxx* y *τzz* los tensores de esfuerzos normales, *τzx* el tensor de esfuerzo tangencial, *μ* y *λ* los parámetros de Lame y *ρ* la densidad. La velocidad compresional esta dada por:

$$a = \sqrt{\frac{l+2m}{r}} \qquad (3)$$

Y la velocidad cortante por:

$$b = \sqrt{\frac{m}{r}} \qquad (4)$$

Este sistema de ecuaciones diferenciales parciales de primer orden es convertido a un esquema "escalonado" o "desplazado" (en inglés "staggered grid") de diferencias finitas, en el que los parámetros y valores de esfuerzos y velocidades (derivadas del desplazamiento) se calculan en una forma intercalada. Siguiendo el artículo de Levander (1988), para el cálculo de derivadas se aplica una aproximación central en las derivadas espaciales de 4to orden y en la derivada temporal de 2do orden.

Las figuras 1 y 2 muestran la ubicación de las velocidades y tensiones, (U_t, W_t, τ_{xx} , τ_{zz}, τ_{zx}) los 3 parámetros utilizados (μ, λ, ρ) en el esquema escalonado. Las esquinas de los cuadrados son los puntos de la grilla: (m,n), (m+1,n), (m+1,n+1), (m.n+1). Los componentes de la velocidad son definidos en los tiempos Δt-1/2 y Δt+1/2, y las tensiones en Δt y Δt+1.

1.2 *Condiciones de estabilidad y dispersión*

Según lo planteado por Levander (1988), para que el algoritmo funcione correctamente, se deben cumplir las siguientes condiciones de estabilidad y dispersión:

1 Estabilidad:

$$\Delta t < \frac{h}{\sqrt{2a\left(c_1 + c_2\right)}}$$

O (5)

$$\Delta t < 0.606h/\alpha$$

Donde c_1 y c_2 son coeficientes de la aproximación de cuarto orden a la primero derivada, (los cuales se calculan con base en método de series de Taylor); h es el mayor delta espacial entre las direcciones x y z y α es la maxima velocidad compresional presente en el modelo.

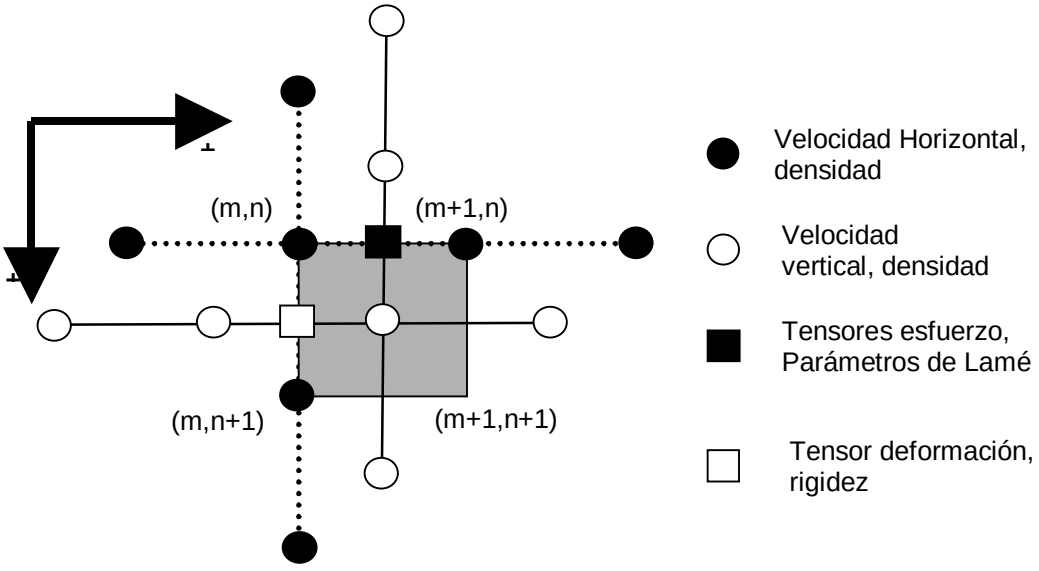


Figura 1 Esquema de la grilla de diferencias finitas y plantilla espacial para la actualización de la velocidad. La plantilla de la velocidad horizontal se muestra como una línea continua, que junto con los nodos de los tensores son utilizados para la actualización. La plantilla de la velocidad vertical se muestra como una línea punteada, que junto con los tensores, son utilizados para la actualización. Es por esto que los procesadores se ven en la necesidad de intercambiar las 2 filas de los tensores para actualizar las velocidades. (Tomado de Levander, 1988)

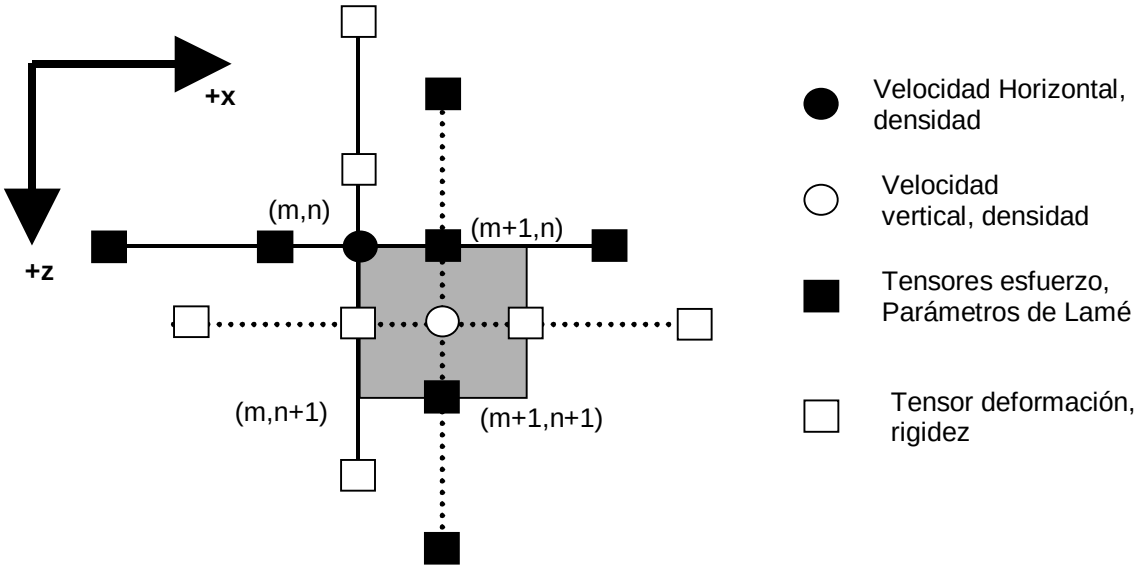


Figura 2. Esquema de la grilla de diferencias finitas y plantilla espacial para la actualización de l los tensores. La plantilla de los tensores de esfuerzo se muestra como una línea continua, que junto con los nodos de los velocidades son utilizados para la actualización. La plantilla del tensor de deformación se muestra como una línea punteada, que junto con las velocidades, son utilizados para la actualización. Es por esto que los procesadores se ven en la necesidad de intercambiar las 2 filas de las velocidades para actualizar los tensores. (Tomado de Levander, 1988)

-Dispersión:

Para disminuir los efectos de dispersión se requiere que existan a lo máximo 5 celdas de la grilla para representar la menor longitud de onda cortante (shear) presente en el modelo, lo que se puede resumir en la siguiente ecuación:

$$l_{\min} = \frac{b_{\min}}{w_c} > 5h$$

Donde ω_c es la frecuencia central de la ondícula, β_{min} es la menor velocidad cortante (shear) presente en el modelo y h es el delta en el espacio mayor entre Δx y Δz.

1.3 Condiciones de Frontera

Dentro de la aplicación desarrollada se pueden escoger las siguientes condiciones de frontera:

1. En la superficie superior se puede aplicar la superficie libre sugerida por Levander.

2. En las fronteras laterales se trabaja con las fronteras absorbentes sugeridas por Higdon (1991), que minimizan el grado de reflexión al aplicar un coeficiente de absorción basados en composiciones de operadores simples de derivadas de primer orden.

3. Paralelización

2.1 .*Descomposición del dominio:*

La descomposición del dominio entre los procesadores se hizo en forma horizontal como lo muestra la figura 3. Cada procesador actualiza los valores de U_t, W_t, τ_{xx} , τ_{zz}, τ_{zx} de su dominio local utilizando las ecuaciones (1) y (2). .El procesador que maneja la

frontera superior aplica superficie libre si es el caso, y cada procesador aplica la frontera de Higdon o Clayton según sea el caso. Los procesadores deben hacer un intercambio de datos, en las fronteras superior e inferior del dominio local, para tener la información necesaria para realizar la actualización de las variables. Este intercambio se debe realizar en cada paso de tiempo.

2.2 *Comunicación y Requerimientos de memoria*

La comunicación juega un papel importante en cualquier aplicación dependiente de la red. Para este caso, la comunicación es un factor crítico, pues en cada paso de tiempo se deben intercambiar las filas entre los procesadores vecinos.

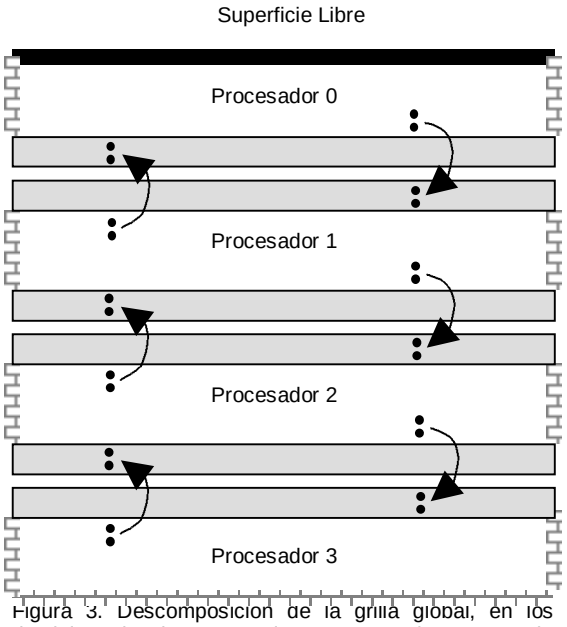


Figura 3. Descomposicion de la grilla global, en los dominios locales que le corresponden a cada procesador. El intercambio entre los procesadores vecinos se debe realizar en cada paso de tiempo.

Se debe entonces encontrar un balance entre la comunicación entre los procesadores y el tiempo que se emplea para ello.

Según lo planteado por Bohlen* (2002) se puede medir la cantidad de datos intercambiados en cada paso de tiempo para el algoritmo como

D= 120*(Nx(p-1))bytes

Donde N es la cantidad de puntos de la grilla en x y p es el número de procesadores en los cuales dividí la grilla.

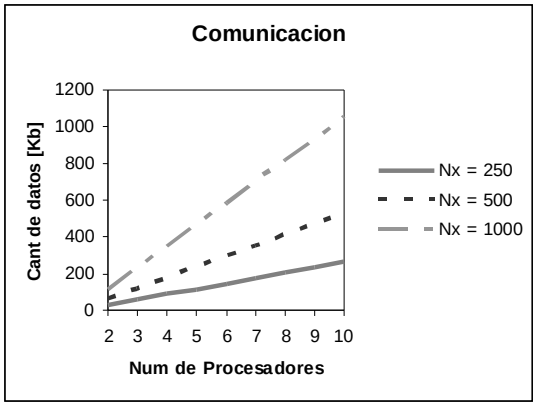


Figura 3. Cantidad de datos que se intercambian en cada paso de tiempo por los procesadores. Este valor depende de Nx . En la grafica se muestra para valores de Nx=250 Nx=500 y Nx=1000

La figura 3 muestra como a medida que aumento el número de procesadores, también aumentó la cantidad de datos a intercambiar entre los procesadores.
Por ejemplo para una simulación de Nx =1000 y una cantidad de procesadores de 0 la cantidad de datos a intercambiar es mayor a 1Mb en cada paso de tiempo.
Por lo tanto para obtener el máximo rendimiento de la aplicación, se recomienda usar en redes de alta velocidad.

La cantidad de memoria utilizada por cada procesador disminuye considerablemente, debido a que cada procesador reserva solo la memoria que necesita, haciendo que se pueda aumentar los tamaños de la grilla. Los requerimientos de memoria se pueden calcular a través de la siguiente fórmula:

$$\text{Memoria / procesador} = \frac{92*Nx*Nz}{p} \text{ bytes}$$

Esta ecuación puede ayudar a determinar cuantos procesadores son necesarios para realizar una simulación de Nx puntos de grilla.

*

2.3 *Aplicación Secuencial Vs Aplicación Paralela*

Para medir la velocidad de la aplicación secuencial se utilizó la ley de Amdahl (Amdahl , 1967) :

$$S = \frac{1}{f + \frac{1-f}{p}}$$

Donde *S* es la ¿? del algoritmo, *p* es el número de procesadores en los cuales se distribuyó el trabajo y *f* es la fracción del algoritmo que corre secuencialmente.

Se realizaron pruebas con 2 modelos geológicos, en un cluster de 5 PCs con sistema operativo Linux Red Hat 9.0 conectados a través de una red Ethernet de 100 Mb, con una memoria RAM de 256 Mb y un procesador de 300 MHz. Para la visualización de los resultados de entrada y salida se utiliza el software de uso libre Seismic Unix SU (Forel et al, 2005), generado y mantenido por el grupo CWP (Center for the Wave Phenomena) de Colorado School of Mines

Los resultados se muestran a continuación.

2.3.1 Demo 1

Se modeló una grilla de 250*250 en la dirección Nz Nx respectivamente. el espaciado del la grilla es 4 m. Se aplicó la condición de superficie libre y frontera Absorbente de Higdon. El tiempo de la simulación fue 2 segundos con un paso de tiempo de 0.001 La información sobre el modelo se puede ver en la figura 4:

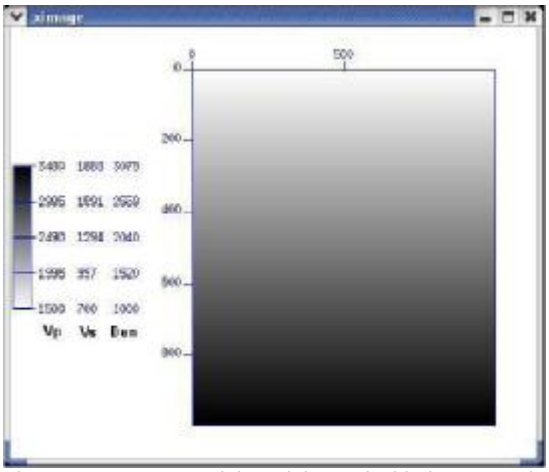


Figura 4. Estructura del modelo, velocidades p, s y la densidad. La aplicación trabaja con archivos de entrada y salida con el formato de SU que hace muy fácil la visualización de los resultados.

En la siguiente gráfica se muestra el tiempo de ejecución contra el número de procesadores utilizados para el modelado.

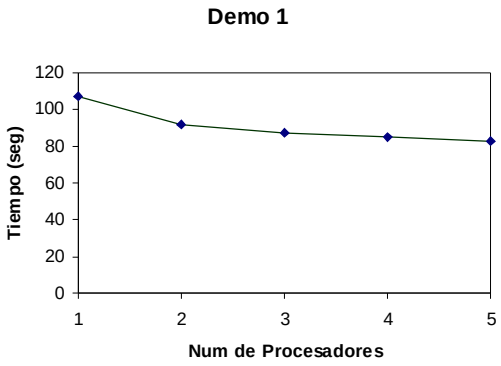


Figura 5. Se observa la comparación de tiempos de la aplicación con respecto a la versión secuencial. Observe que la reducción de tiempo con respecto a la aplicación secuencial es pequeño.

Se puede observar de la gráfica que la reducción de tiempo para este modelo en especial no es muy significante, esto debido que la simulación es muy pequeña. Este ejemplo nos hace pensar que los modelos que se deben ejecutar en una aplicación paralela deben ser suficientemente grandes, para que se note una reducción considerable en el tiempo.

*

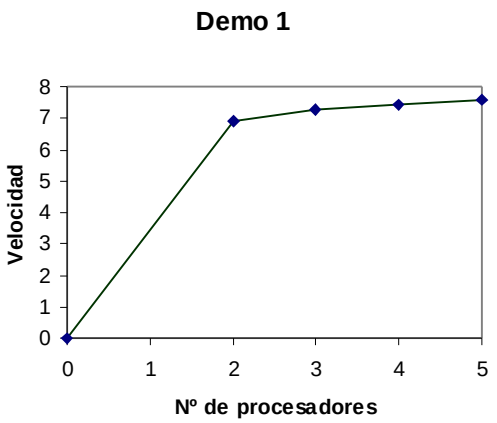


figura 6. La aceleración para el demo1. Observe que a medida que se agregan procesadores la aceleración va creciendo.

La aceleración para el demo1 va en aumento, pero no es el esperado. Esto se debe a que el tiempo disminuyó (Como lo muestra en la figura 5) pero no de forma dramática. Ahora, el tiempo total también se ve afectado por la comunicación en la red, y como se mencionaba anteriormente, el cluster donde se realizaron las pruebas no esta conectado con una red de alta velocidad.

2.3.2 Demo 2

Se modeló una grilla de 500*500 en la dirección Nz Nx respectivamente. el espaciado del la grilla es 2 m. Se aplicó la condición de superficie libre y frontera disipativa. El tiempo de la simulación fue 2 segundos con un paso de tiempo de 0.0002 La información sobre el modelo se puede ver en la figura 7.

En la visualización de los tiempos de ejecución (figura 8) se puede ver que se reducen casi a la mitad. Debido que la grilla es mucho mas grande y el tiempo de simulación es 100 veces mas grande que el modelo anterior.
En la figura 9 se observa la relación entre los procesadores y la aceleración

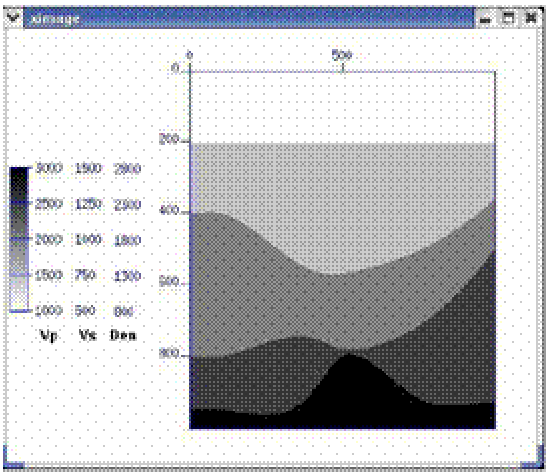


Figura 8. La estructura del modelo, velocidades p, s y la densidad.

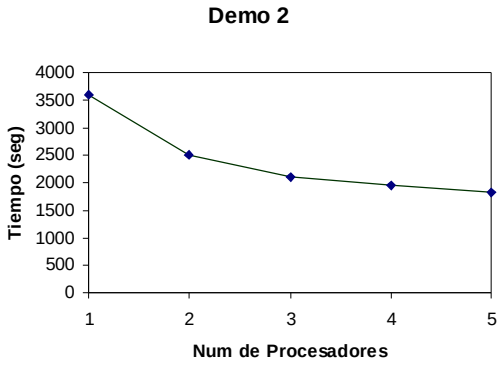


Figura 9. Comparación de tiempos de ejecución a aplicación secuencial y aplicación paralela hasta 5 nodos.

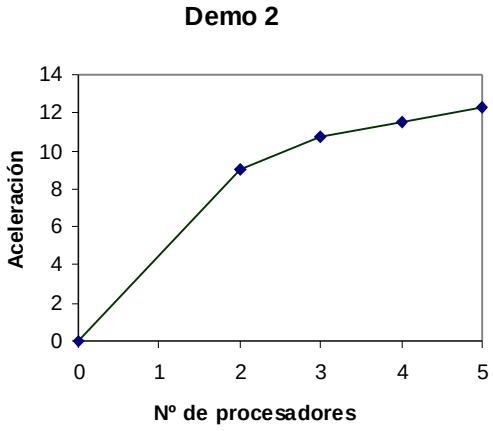


Figura 10 La aceleración para el demo 2

2.4 *Visualización de Resultados*

*

Los resultados que genera EcoElas2D son archivos binarios con las matrices de números flotantes que tienen los resultados, buscando compatibilidad con librerías como la del SU (Seismic Unix) y la del SEP (Stanford Exploration Project).

Para el demo2 los resultados se ilustran en las Figuras 12 y 13.

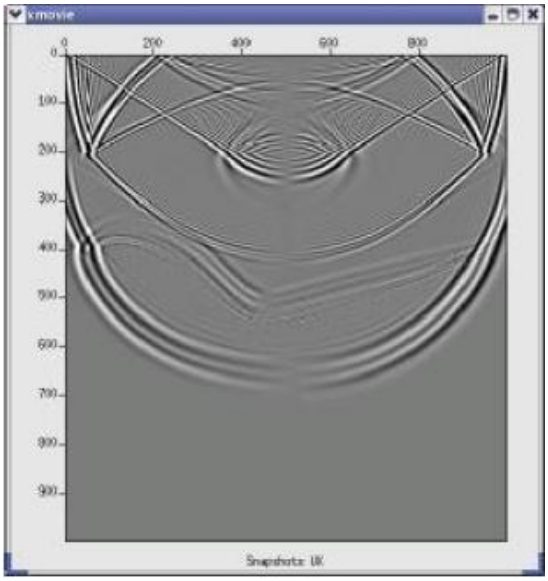


Figura 12. Snapshot de la componente horizontal para el demo2.

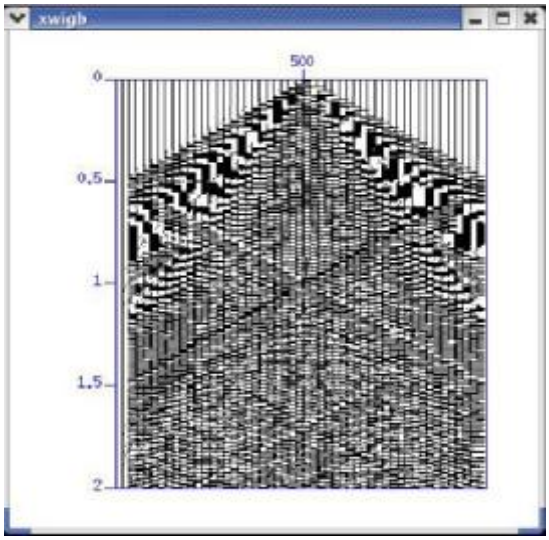


Figura 13. Sismograma sintético de la componente horizontal para el demo2

4. Desarrollos adicionales y posibles desarrollos futuros

Adicionalmente a la versión del software presentada hasta aquí, se ha desarrollado una versión que tiene la posibilidad de representar una superficie topográfica rugosa y de tener tamaño variable de la celda (Hayashi et al., 2000).

Esta última característica permite adaptar el tamaño de la grilla a los requerimientos de la velocidad y longitud de onda críticas. Como se vio anteriormente, estos requerimientos afectan la estabilidad y dispersión de la onda resultante, y en general son críticos solo en una parte pequeña del modelo, (en particular en estratos de baja velocidad), de manera que tener la posibilidad de un tamaño adecuado de grilla para cada sector permite disminuir la exigencia de recursos de computo. Aunque esta versión no se ha implementado en forma paralela, sería una necesidad futura, de manera que de esta forma sea aún más adecuado para utilizar el software en modelos realistas con un consumo razonable de recursos de computador.

El método de diferencias finitas tiene un gran potencial para su aplicación a estudios de la propagación de ondas. En principio es posible implementar cualquier modelo de propagación de las ondas, estando su límite más en la tecnología en computación. Los avances en poder de los computadores hacen factible la implementación de algoritmos cada vez más exactos. Es así como se han desarrollado métodos 3D y que tienen modelos viscoelásticos (e.g. Bohlen, 2002). También es posible implementar modelos anisotrópicos.

El desarrollo de este tipo de modelos más complejos y exactos es una posibilidad para desarrollos futuros.

5. Conclusiones

Los ejemplos y las experiencias anteriores demuestran que el esquema de diferencias finitas escalonado es eficiente.

En este artículo se muestra la efectividad de la implementación del código paralelo a través de la interfaz de paso de mensajes (MPI) , haciendo una división del dominio a lo largo del eje Z. Realizando la descomposición de esta manera, evito mayor

*

comunicación entre los procesadores haciendo mas eficiente el algoritmo.

Las pruebas realizadas muestran que para una mayor eficiencia de EcoElas2D se debe ejecutar en redes de alto desempeño, o bien a través de maquinas creadas para estos propósitos.

Sin embargo la utilización de Clusters Linux, de costo relativamente bajo, es una importante opción a tener en cuenta en un futuro.

Como trabajo futuro se plantea la aplicación de esta técnica a la segunda versión de la aplicación, donde se manejan don tipos de grilla, así como fronteras irregulares.

También se plantea la posibilidad de trabajar con modelos más complejos, incluyendo modelos 3D, viscoelásticos y anisotrópicos.

Bibliografía

Amdahl, G 1967. The validity of the single processor approach to achieving large scale computing capabilities. En: AFIPS Conference Proceedings, Spring Joint Computing Conference, Vol 30, 483-485.

Bohlen Thomas. Parallel 3-D viscoelastic finite difference seismic modeling. Computers & Geosciences 28(2002) 887-889.

Clayton, R and Engquist, Absorbing boundary conditions for acoustic and elastic wave equations: Boletín Sociedad Americana de Sismolgia, 67 1529-1540

Forel, D., Benz, T., y Pennington, W. 2005. Seismica data processing with seismic Un*x. SEG Course Notes Series, Tulsa, USA.

Hayashi, K., Burns, D. y Toksoz, N. 2000. Discontinuous-grid finite-difference seismic modeling including surface topography. Bulletin of the seismological society of America, v. 91, n. 6, pp 1750-1764.

Higdon, R. L. Absorbing boundary conditions for elastic waves. Geophysics, 56(1), 231-241.

Levander, Alan R. Fourth-orden finite difference P_SV seismograms. Geophysics 53(11), 1425-1436

Virieux, Jean, 1986, P-SV wave propagation in heterogeneous media: Velocity-stress finite-difference method, Geophysics 51, 889-901 ",

*

ANEXO B
MANUAL DE USUARIO

EcoElas2D Versión Paralela

MODELAMIENTO DE PROPAGACIÓN DE ONDAS ELÁSTICAS 2D EN CUERPOS SÓLIDOS USANDO EL METODO DE DIFERENCIAS FINITAS

Manual de Usuario

INSTITUTO COLOMBIANO DEL PETROLEO
ECOPETROL-ICP
Piedecuesta, Agosto del 2006

Índice de contenido

1 INTRODUCCIÓN 4

2 MARCO TEÓRICO 4

3 DESCRIPCIÓN DEL PROGRAMA 6

3.1 CARACTERÍSTICAS 6
3.2 INFORMACIÓN TÉCNICA 6
3.3 ENTRADAS 6
3.4 SALIDAS 8
3.5 DESCRIPCIÓN DE LAS PRINCIPALES OPCIONES 8
3.5.1 Tipo de Fuente (source_type) 8
3.5.2 Tipo de Ondícula (wavelet_type) 10
3.5.3 Superficie Libre (fs) 10
3.5.4 Suavizado (smooth) 11
3.5.5 Fronteras Disipativas (taper) 59
3.5.6 Fronteras Absorbentes (higdon) 13
3.5.7 Snapshots y Ssísmica de Presión Calculada 13

4 TUTORIAL 13

4.1 INSTALACIÓN 13
4.2 CREACIÓN DEL MODELO 14
4.3 CREACIÓN DE LOS ARCHIVOS DE VELOCIDADES 17
4.4 DEFINICIÓN DE LOS PARÁMETROS 18
4.5 EJECUCIÓN DE ECOELAS2D 19
4.6 VISUALIZACIÓN DE LOS RESULTADOS 19
4.6.1 Usando el SU (Seismic Unix) 19
4.6.2 Usando el SEP (Stanford Exploration Project) 22

5 EJEMPLOS DE RESULTADOS 23

5.1 DEMO1 23
5.2 DEMO2 25
5.3 DEMO3 27

6 CRÉDITOS 29

Introducción

EcoElas2D es un programa de modelamiento de la propagación de ondas elásticas 2D en cuerpos sólidos usando el método de las diferencias finitas, que corre en una máquina paralela, como una maquina secuencial.

El presente documento presenta una breve descripción sobre sus principales características y forma de uso.

Para la realización de este manual se tuvo como base el manual creado por NUMERICA LTDA para EcoElas2D 1.0.

Marco Teórico

EcoElas2D esta basado en la aproximación en diferencias finitas de la ecuación de onda elástica presentada por Alan Levander en el artículo **"Fourth-order finite-difference P-SV seismograms",Geophysics 53, 1425-1436 – 1988.**

Formulación:
En un sistema cartesiano 2D con *X* como eje horizontal y positivo hacia la derecha y *Z* como eje vertical positivo hacia abajo, la ecuación de movimiento de ondas P-SV esta dada por:

$$r\,\frac{\partial Ut}{\partial t} = \frac{\partial txx}{\partial x} + \frac{\partial txz}{\partial z}$$

y

$$r\,\frac{\partial Wt}{\partial t} = \frac{\partial tzx}{\partial x} + \frac{\partial tzz}{\partial z}$$

y las leyes constitutivas para un medio isotrópico son:

$$txx = (l + 2m)\frac{\partial U}{\partial x} + l\,\frac{\partial W}{\partial z}\ ,$$

$$tzx = m\left(\frac{\partial U}{\partial z} + \frac{\partial W}{\partial x}\right),$$

y

$$tzz = (l + 2m)\frac{\partial W}{\partial z} + l\,\frac{\partial U}{\partial x}$$

donde *U* y *W* son los componentes del desplazamiento en *x* y *z, Ut* y *Wt* son las velocidades de las partículas, τ_{ij} son los esfuerzos, λ y μ son los parámetros de Lame siendo μ la rigidez. La velocidad compresional está dada por $a = \sqrt{(l + 2m)/r}$ y la velocidad cortante por $b = \sqrt{m/r}$, donde ρ es la densidad.

De la aproximación en diferencias finitas de las anteriores ecuaciones presentadas en el artículo en referencia, se establecen los criterios de estabilidad y dispersión que maneja EcoElas2D y están dados por las siguientes condiciones:

- **Estabilidad :**

$$\Delta t < \frac{h}{\sqrt{2a\,(c_1 + c_2)}}$$

o

$$\Delta t < 0.606(h/a)$$

Donde c_1 y c_2, son los coeficientes entrante y saliente de la aproximación de cuarto orden a la primera derivada, *h* es el mayor delta en *x* y *z*, y α es la mayor velocidad compresional presente en el modelo.

- **Dispersión:**

Para disminuir los efectos de dispersión se requiere que existan a lo máximo 5 celdas de la grilla para representar la menor longitud de onda cortante (shear) presente en el modelo, lo cual lo podemos resumir en la siguiente ecuación:

$$v_c >= \frac{b_m}{5h}$$

Donde v_c es la frecuencia central de la ondícula, b_m es la menor velocidad cortante (shear) presente en el modelo y *h* es el delta en el espacio, el mayor entre x y z.

Para la característica de fronteras absorbentes se utilizó el esquema propuesto por Robert L. Higdon en el artículo **"Absorbing boundary conditions for elastic waves" Geophysics 56, 231-241 – 1991.**

Algunas de las características que se implementaron en EcoElas2D fueron ideas tomadas del software **Ela2D** de Joachim Falk (Ver: Hard- & Software of AGG Hamburg -> http://www.agg.dkrz.de/soft y Falk, J., Tessmer, R., Gajewski, D., 1996, Tube Wave Modeling by the Finite-Difference Method with varyng Grid

Spacing, Pure and Applied Geophysics)

Descripción del Programa

Características

- Solución directa de la ecuación de Onda elástica.
- Operadores de Diferencias Finitas de 4to Orden para derivadas en el espacio.
- Esquema de 2do Orden de Diferencias Finitas en el Tiempo.
- Condición de Fronteras absorbentes de Higdon + Fronteras Disipativas.
- Superficie Libre o Frontera absorbente en la frontera superior de la grilla .
- Utiliza el mismo esquema y estándares utilizados en los programas del SU (Seismic Unix).

Información Técnica

- Desarrollado en Lenguaje C.
- Utiliza los estándares de la GNU lo que lo posibilita instalarse y ejecutarse en la mayoría de Unix.
- Documentación del código siguiendo los estándares de Doxygen[4].
- Librería usada para la paralelización LAM-MPI2.
- El requerimiento de maquina depende del tamaño de la grilla que se quiere modelar y el tiempo deseado de solución.

Entradas

El programa como entrada recibe archivos con las Velocidades P, S y las densidades además de los parámetros que se resumen a continuación:

Parámetro	**Descripción**
np	*Numero de procesadores utilizados para la ejecución del programa*
Nx=100	*cantidad de celdas en X*
Nz=100	*cantidad de celdas en Z*
Dx=0.01	*delta en X*

[4] Doxygen. Es un sistema de documentación de códigos muy popular en el software GNU, que permite generar la documentación en línea de una forma ordenada y esquematizada (graficos de relaciones) de una manera automática, simplemente siguiendo algunas reglas al momento de colocar los comentarios dentro del código fuente.

Parámetro	Descripción
Dz=0.01	*delta en Z*
Dt=0.01	*delta en el tiempo (seg)*
fcent=30.0	*frecuencia central de la ondicula (Hz)*
Vp=vp.bin	*archivo que contiene las velocidades P*
Vs=vs.bin	*archivo que contiene las velocidades S*
den=density.bin	*archivo que contiene las densidades*
pgeof=pgeof.txt	*archivo que contiene la posición de los geofonos*
par=data.in	*archivo que contiene parámetros a usar*
tmax=1.0	*tiempo de simulación (seg)*
*spx=(nx*dx)/2*	*posición de la fuente en X*
spz=0	*posición de la fuente en Z*
source_type=1	*tipo de fuente* *1 -> dilatacional point source* *2 -> vertically oriented single couple* *3 -> horizontally oriented single couple*
wavelet_type=1	*tipo de ondicula* *1 -> 2nd derivative of Gauss-function* *2 -> Kosloff wavelet* *3 -> Ricker wavelet* *4 -> Kuepper wavelet*
seismic=1	*determina si graba sismogramas (0=no 1=si)*
step_seismic=0.3	*determina cada cuanto tiempo (seg) graba el sismograma*
snapshots=1	*determina si desea grabar snapshot (0=no 1=si)*
delay_ini=dt	*tiempo de espera para empezar a grabar snapshots (seg)*
*step_snapshots=5*dt*	*cada cuanto tiempo (seg) graba el un snapshot*
higdon=1	*determina si aplica fronteras absorbentes (0=no 1=si)*
higdon_width=2	*ancho de la frontera absorbente*
fs=1	*determina si aplica superficie libre (0=no 1=si)*
smooth=0	*determina si aplica suavizado a las velocidades y densidades (0=no 1=si)*
taper=0	*determina si aplica fronteras disipativas (0=no 1=si)*
tw=1	*ancho de la frontera disipativa*
pr=0	*porcentaje de disipación*
wg=10e5	*ganancia dada al wavelet*
file_snapx=SnapX.bin	*archivo de salida de los Snapshots en X. Si el usuario utiliza como nombre "null" no se*

Parámetro	Descripción
	escribirá este archivo
file_snapz=SnapZ.bin	*archivo de salida de los Snapshots en Z. Si el usuario utiliza como nombre "null" no se escribirá este archivo*
file_snappr=SnapPR.bin	*archivo de salida de los Snapshots Presión. Si el usuario utiliza como nombre "null" no se escribirá este archivo*
file_seismicx=SeismicX.bin	*archivo de salida de la sísmica en X*
file_seismicz=SeismicZ.bin	*archivo de salida de la sísmica en Z*
file_seismicpr=SeismicPR.bin	*archivo de salida de la sísmica Presión*
file_wavelet	*Wavelet.bin*
verbose=1	*determina si imprimir resultados cada dt file_out=data.out archivo de salida del reporte de ejecución*

Salidas

El programa como salida da los siguientes Archivos:

- SnapX.bin SnapShots de los desplazamientos en X.
- SnapZ.bin SnapShots de los desplazamientos en Z.
- SnapPR.bin SnapShots de la presiones calculadas.
- SeismicX.bin Sísmica de los desplazamientos en X.
- SeismicZ.bin Sísmica de los desplazamientos en Z.
- SeismicPR Sísmica de las presiones calculadas.
- Wavelet.bin Ondicula utilizada en el modelamiento.
- Data.out Archivo ascii con el reporte de ejecución.

Descripción de las Principales Opciones

En esta sección se describirá como funciona internamente el programa al utilizar las principales opciones que transforman considerablemente los resultados del modelamiento.

Tipo de Fuente (source_type)

El tipo de fuente es un entero de 1 a 3 y puede ser de los siguientes que se describen a continuación:

1 : dilatational point source: Al utilizar este tipo de fuente el programa ejecuta las siguientes acciones:

- Escala los valores de la Ondícula de la siguiente forma:

$$W(i) = W(i) * 1.0e7 * \sqrt{3l + m}$$

Donde *W(i)* es el valor de la Ondícula calculado para el tiempo *i*Dt*, *0 < i < Longitud de la Ondícula*, λ y μ son los parámetros de siendo μ la rigidez para la celda donde se encuentra la fuente.

- Mientras el tiempo de modelamiento sea menor que la longitud de la Ondícula, modifica los tensores de la siguiente forma:

Txx[spz][spx]=Txx[spz][spx]+W[t] ,

Tzz[spz][spx]=Tzz[spz][spx]+W[t]

Donde *Txx y Tzz* son los tensores de esfuerzo normal en x y z respectivamente, *spz* y *spx* son los índices de la celda donde se colocó la fuente, y *W[t]* es el valor de la ondícula en el tiempo t del modelamiento, *0 < t < Tiempo Máximo de Modelamiento.*

2 : vertically oriented single couple: Al utilizar este tipo de fuente el programa ejecuta las siguientes acciones:

- Mientras el tiempo de modelamiento sea menor que la longitud de la Ondícula, le suma al desplazamiento en Z de la celda donde se colocó la fuente el valor de la Ondícula en ese tiempo de la siguiente forma:

Uz[spz][spx]=Uz[spz][spx]+W[t]

Donde *Uz* son los desplazamientos en Z, *spz* y *spx* son los índices de la celda donde se coloco la fuente, y *W[t]* es el valor de la ondícula en el tiempo t del modelamiento, *0 < t < Tiempo Máximo de Modelamiento.*

3 : horizontally oriented single couple: Al utilizar este tipo de fuente el programa ejecuta las siguientes acciones:

- Mientras el tiempo de modelamiento sea menor que la longitud de la Ondícula, le suma al desplazamiento en X de la celda donde se colocó la fuente

el valor de la Ondícula en ese tiempo de la siguiente forma:

Ux[spz][spx]=Uz[spz][spx]+W[t]

Donde *Ux* son los desplazamientos en X, *spz* y *spx* son los índices de la celda donde se coloco la fuente, y *W[t]* es el valor de la ondícula en el tiempo t del modelamiento, *0 < t < Tiempo Máximo de Modelamiento.*

Tipo de Ondícula (wavelet_type)

El tipo de ondícula es un entero de 1 a 4 y puede ser de los siguientes que se describen a continuación:

1 : 2nd derivative of Gauss-function

2 : Kosloff wavelet

3 : Ricker wavelet

4 : Kuepper wavelet

Cada una de estas ondículas se calculan según las ecuaciones definidas para cada tipo. El código de implementación de estas funciones se puede ver en el archivo “wavelet.c”. El programa además también salva a un archivo binario la ondícula para que el usuario posteriormente la pueda visualizar.

Superficie Libre (fs)

EcoElas2D puede tomar la frontera superior (la de la superficie) de tres formas diferentes como se describe a continuación:

Frontera Absorbente:

Trabaja de esta forma cuando el parámetro *fs* es igual a 0 y *higdon* es igual a 1. Cuando está de este modo el programa trata de absorber la ondícula como se describe mas adelante en la sección de fronteras absorbentes.

Superficie Fija:

Se activa cuando el parámetro *fs* es igual a 0 y *higdon* es igual a 0. Cuando está de esta manera, los desplazamientos en toda la frontera superior se hacen igual a 0 con lo cual se provocará una reflexión grande en la superficie.

Superficie Libre:

Se establece en este modo cuando el parámetro *fs* es igual a 1 sin importar el

valor que tomen los otros parámetros. Cuando trabaja de este modo para todas las celdas de la superficie se hace la siguiente operación:

Txz[1][j]=0.0 ,
Tzz[1][j]=0.0

Donde *Tzz* es el tensor de esfuerzo normal en z y *Txz* es el tensor cortante, *j* es el índice para todas las celdas de la superficie, donde *1 < j < Nx (numero de celdas en x).*

Suavizado (smooth)

El suavizado puede tener un valor de 0 a 2 como se describe a continuación:

0 : No Suavizado: Cuando este parámetro es igual a 0 no se realiza ningún tipo de suavizado sobre los datos.

1 : harmonic averaging of shear modulus : Cuando se usa este tipo de suavizado se realiza las siguientes operaciones sobre los datos:

$$m(i,j) = \frac{4}{\frac{1}{m(i,j)} + \frac{1}{m(i,j+1)} + \frac{1}{m(i+1,j)} + \frac{1}{m(i+1,j+1)}},$$

$$r(i,j) = \frac{r(i,j) + r(i,j+1)}{2}$$

$$r2(i,j) = \frac{r(i,j) + r(i+1,j)}{2}$$

donde μ es el parámetro de Lame correspondiente a la rigidez y ρ es la densidad dada como entrada en el modelo, ρ2 es la misma densidad pero promediada hacia delante en las x para ser utilizada en la derivadas en x negativas, *i,j* son los índices para recorrer la grilla en x y z respectivamente.

2 : slowness averaging: Este tipo de suavizado realiza las siguientes operaciones sobre los datos:

$$l(i,j) = \frac{4}{\frac{1}{l(i-1,j-1)} + \frac{1}{l(i+1,j-1)} + \frac{1}{l(i-1,j+1)} + \frac{1}{l(i+1,j+1)}},$$

$$m(i,j) = \frac{4}{\frac{1}{m(i,j)} + \frac{1}{m(i,j+1)} + \frac{1}{m(i+1,j)} + \frac{1}{m(i+1,j+1)}}$$

$$r(i,j) = \frac{r(i,j) + r(i,j+1)}{2}$$

$$r2(i,j) = \frac{r(i,j) + r(i+1,j)}{2}$$

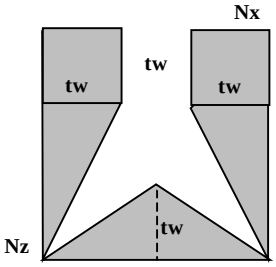
donde μ y l son los parámetros de Lame siendo μ la rigidez y ρ es la densidad dada como entrada en el modelo, ρ2 es la misma densidad pero promediada hacia delante en las x para ser utilizada en la derivadas en x negativas, *i,j* son los índices para recorrer la grilla en x y z respectivamente.

Fronteras Disipativas (taper)

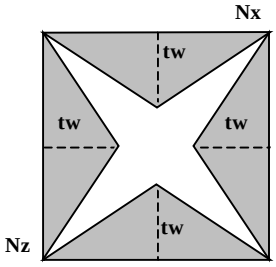
Esta característica se activa cuando el parámetro *taper* es igual a 1 y actúa con otros 2 parámetros que son *tw* y *pr.* El parámetro *tw* define el ancho de la zona de frontera en metros, que debería ser aproximadamente 2 longitudes de onda; el punto donde se ubica la fuente debería estar aproximadamente alejado 2 longitudes de onda de esta zona. El parámetro *pr* es el factor de escalamiento que podría dejarse siempre en 25.

Cuando está activa la característica de fronteras disipativas, el programa multiplica por una función exponencial todos los desplazamientos tomando la siguiente forma:

Superficie Libre Activada (fs=1):



Superficie Libre Desactivada (fs=0):



Donde el área sombreada corresponde a las celdas de la grilla que se multiplicaran por la función exponencial, y *tw* es el numero de celdas que equivalen al ancho de la frontera disipativa dado por el usuario en el parámetro *tw*.

Fronteras Absorbentes (higdon)

La característica de fronteras absorbentes se activa cuando el parámetro *higdon* es igual a 1, y utiliza un parámetro adicional denominado *higdon_width* que especifica el número de filas o columnas a las que se le aplicará la frontera absorbente. La metodología empleada para esta característica está basada en el articulo “Absorbing boundary conditions for elastic wave” cuyo autor es Robert L. Higdon (GEOPHYSICS, VOL 56, No 2 February 1991), P231-241.

Snapshots y Ssísmica de Presión Calculada

El programa como salida a parte de los snapshots y sismica de los desplazamientos en X y Z, también da un snapshot y sísmica de una presión calculada a partir de los tensores de esfuerzo de la siguiente manera:

$$\Pr = \frac{txx + tzz}{\frac{l + 2m}{r}}$$

donde τ_{xx} y τ_{zz} son los tensores de esfuerzo normales, λ y μ son los parámetros de Lame siendo μ la rigidez y ρ es la densidad.

Tutorial

Instalación

Para compilar e instalar "EcoElas2D Versión Paralela" se debe definir la ruta donde se va a instalar, en una variable de sesión de la siguiente forma:

$ ECOELASPATH=/ruta/instalar

$ export ECOEASPATH

se debe ejecutar los siguientes comandos desde el directorio donde se descomprimió el archivo de distribución con los códigos fuente de la aplicación:

$ ./configure

$ make

$ make install

Lo anterior instala el programa "ecoelas2dp" en el directorio "bin" que se encuentra en el directorio desde el cual se ejecuto la instalación. Si desea instalarlo en otro directorio debe ejecutar los anteriores comandos con los siguientes cambios:

$ ./configure --prefix=[Directorio donde se desea instalar]

$ make

$ make install

Por ultimo se debe añadir en la variable de entorno “PATH” la ruta del directorio donde quedó instalado EcoElas2d.

Creación del Modelo

Para el modelamiento lo primero que se requiere es el modelo a simular. Para lo anterior se desarrolló una aplicación gráfica complementaria que permite generar los archivos ascii con las coordenadas del modelo siguiendo los estándares del SU (Seismic Unix) como se describe a continuación:

La aplicación se llama “modelo” y se encuentra bajo el directorio con el mismo nombre dentro del directorio donde se instaló EcoElas2D Versión Paralela. Luego de ejecutarlo aparecerá la ventana que se muestra a continuación:

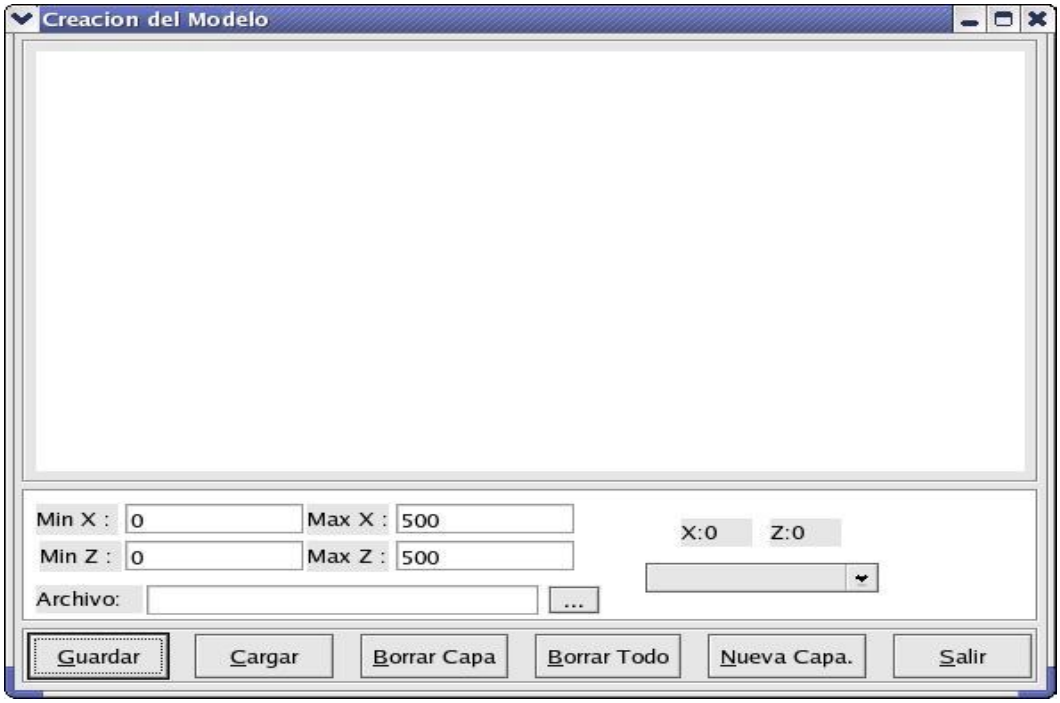


Figura 1. Ventana principal.

Esta es la ventana que se muestra cuando se inicia la aplicación, como parámetros iniciales toma por defecto los límites Min X=0 , Max X=500, Min Z=0 y Max Z=500.

Para añadir una nueva capa debemos hacer clic sobre el botón etiquetado con "*Nueva Capa*" este nos dibujará una capa horizontal que va desde la mitad del rango de *Z* y *X=0* hasta *X=Max*. Teniendo esta capa se puede ubicar los otros puntos o trasladar esta capa ya sea mas arriba o mas abajo como lo desee el usuario. (Ver figura 2a, 2b).

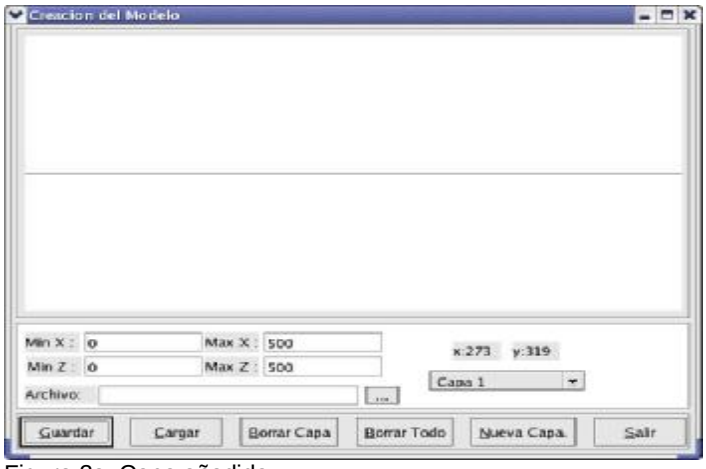


Figura 2a. Capa añadida

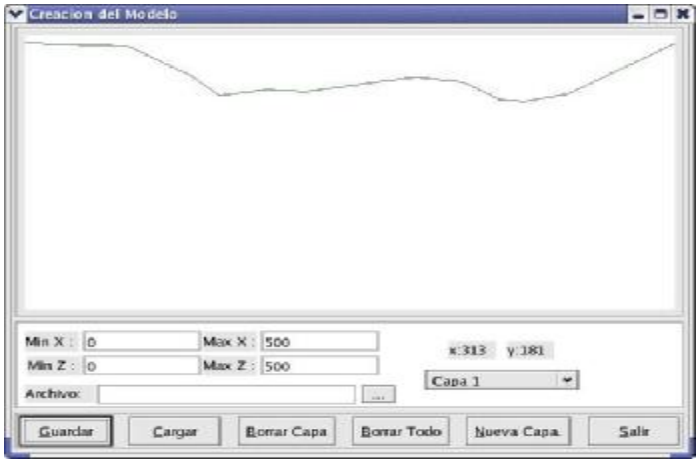


Figura2b. Capa modificada

Al añadir una nueva capa al cuadro de lista de capas se le inserta una nueva etiqueta de la capa añadida la cual puede ser seleccionada en cualquier momento para permitir la edición de dicha capa, o el borrado de la misma haciendo clic en el botón "*borrar capa*".

Se puede seguir añadiendo las capas que se desee y modificando su geometría de acuerdo al modelo que se quiera representar.

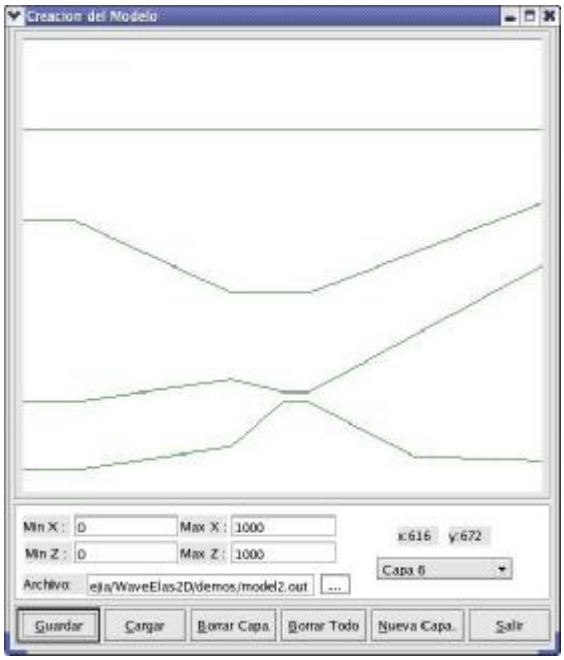


Figura 3. Modelo de varias capas

Si el usuario desea guardar dicho modelo solo necesita escribir el nombre del archivo que desea guardar en la caja de texto etiquetada con Archivo y hacer clic en el botón guardar y le será mostrado el mensaje de archivo guardado satisfactoriamente. (Figura 4).

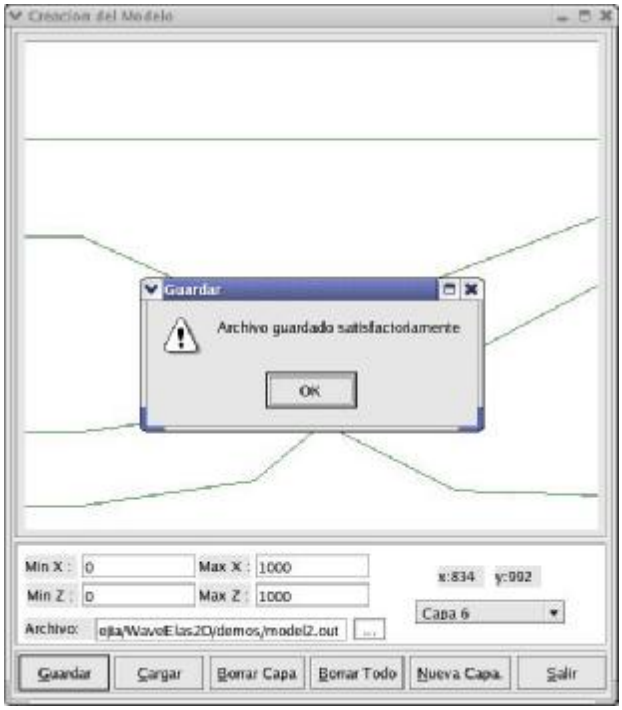


Figura 4. Archivo guardado satisfactoriamente

El archivo generado tiene la siguiente forma según los estándares del SU.

```
0       0
1000    0
1       -99999
0       200
1000    200
1       -99999
0       400
100     400
400     560
550     560
1000    360
1       -99999
0       800
100     800
400     750
500     780
550     780
1000    500
1       -99999
0       950
100     950
400     900
500     800
550     800
750     920
1000    930
1       -99999
0       1000
1000    1000
1       -99999
```

Creación de los Archivos de Velocidades

Una vez se tiene el archivo con el modelo, se procede a crear los archivos de velocidades haciendo uso del programa “unif2” del Su de la siguiente forma:

```
$ unif2 < modelo.out nx=250 dx=2 nz=250 dz=2 \ v00=1000,1500,2000,2500,3000

> velp.bin method=spline
```

Este comando nos generará el modelo de velocidades P en el archivo “velp.bin”.

Este modelo se puede visualizar mediante la utilidad “ximage” del SU con el comando:

ximage < velp.bin n1=250 d1=2 legend=1 title=”Velocidades P” &

donde *n1* es el número de celdas en *Z* y *d1* el delta de *Z*, *n2* y *d2* el número de celdas en *X* y su respectivo *dx*. Esto mostrará la siguiente imagen:

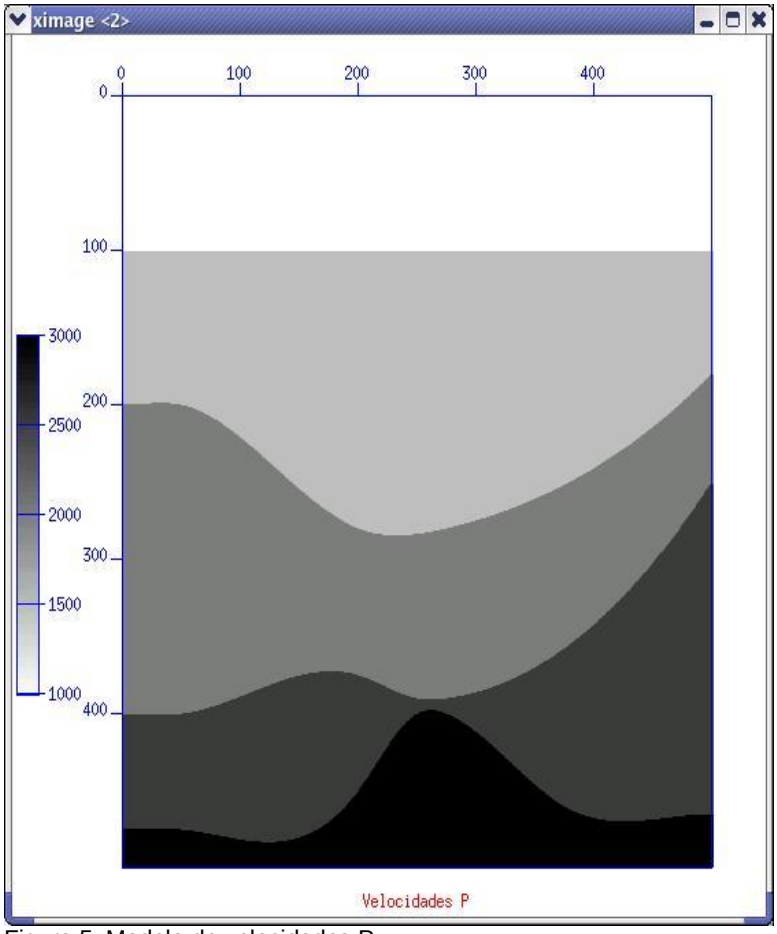


Figura 5. Modelo de velocidades P.

De la misma forma se crea el archivo de Velocidades S y densidades cambiándole el valor del parámetro V00

Definición de los parámetros

Los parámetros de la misma manera que los programas del SU se pueden dar en la línea de comandos o mediante un archivo como se muestra a continuación:

ecoelas2dp np= nx= nz= dx= dz= dt= fcent= vp= vs= den= pgeof= [Param. Opcionales]

o ",
ecoelas2dp np= par=[Archivo Ascii con todos los parametros requeridos y opcionales]

*

Ejecución de EcoElas2D

Una vez se ejecuta el programa, este muestra en pantalla valores como la máxima Velocidad P encontrada y la mínima Velocidad S encontrada, así como también mensajes de advertencia o error cuando se violan los criterios de dispersión o estabilidad. Si el parámetro “verbose” es igual a 1 (valor por defecto) se mostrara el valor absoluto de las máximas amplitudes encontradas para cada SnapShot, así como el numero de ciclos que se han ejecutado. Toda esta información sera encontrada en el nodo maestro, del cluster que sea utilizado.

El reporte de todos estos mensajes quedaran en el archivo “data.out” o en el que el usuario especifique en los parámetros.

El primer paso anteres de ejecutar EcoElas2D es verificar que el ambiente LAM-MPI este ejecutandose en su cluster. Para compobarlo ejecute ecoelas2dp y le mostrara un mensaje en donde le informará que el ambiente no esta instanciado.

Paa “levantar” el ambiente LAM-MPI solo debe ejecutar la siguiente instrucción:

lamboot [archivonombrehosts]

Este archivo debe contener las direcciones ip de los equipo o nombres que serán los nodos

Ejemplo:

192.168.19.12 nodo1

192.168.19.13 nodo2

192.168.19.14 nodo3

Si usted no especifica un np (número de procesadores) cuando ejecute ecoelas2dp el asumirá que usted desea utlizar todos los equipos que se encuentran en el archivo.

Visualización de los Resultados

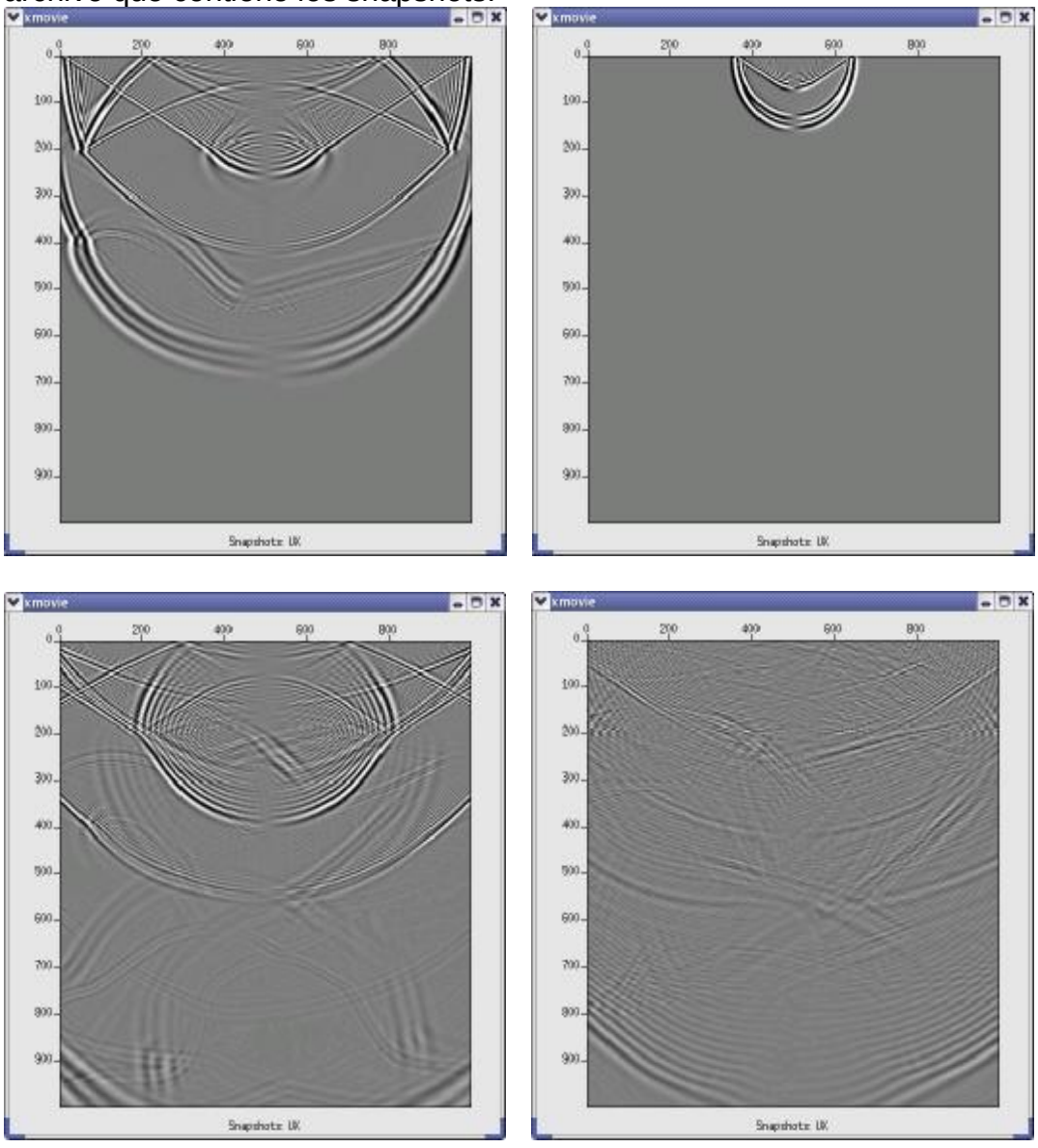
Los resultados que genera EcoElas2D son archivos binarios con las matrices de números flotantes que tienen los resultados, buscando compatibilidad con librerías como la del SU (Seismic Unix) y la del SEP (Stanford Exploration Project). A continuación se muestran los esquemas de cómo visualizar los resultados haciendo uso de cualquiera de estas dos librerías.

Usando el SU (Seismic Unix)

Para visualizar los resultados usando el SU se pueden utilizar un flujo como el que se muestra a continuación:

Visualización de los SnapShots:
suaddhead < SnapX.bin ns=$nz |
*suxmovie clip=1.0 loop=1 n1=500 d1=2 n2=500 d2=2 f2=0.0 dt=0.0002 width=500 height=500 *
title="Snapshots UX" &

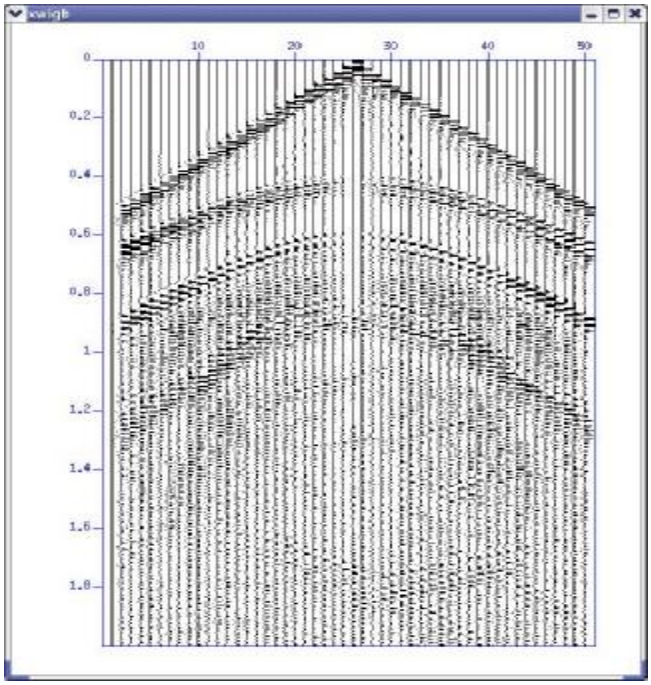
donde ns y n1 es el número de celdas en Z, d1 es el delta de Z, n2 es el número de celdas en X y d2 es el delta de X, dt es el delta de tiempo, SnapX.bin es el archivo que contiene los snapshots.



Visualización de la Sismica:
suaddhead < SeismicX.bin ns=1001 |

suxwigb clip=1.0 n1=1001 d1=0.002 dt=0.0003

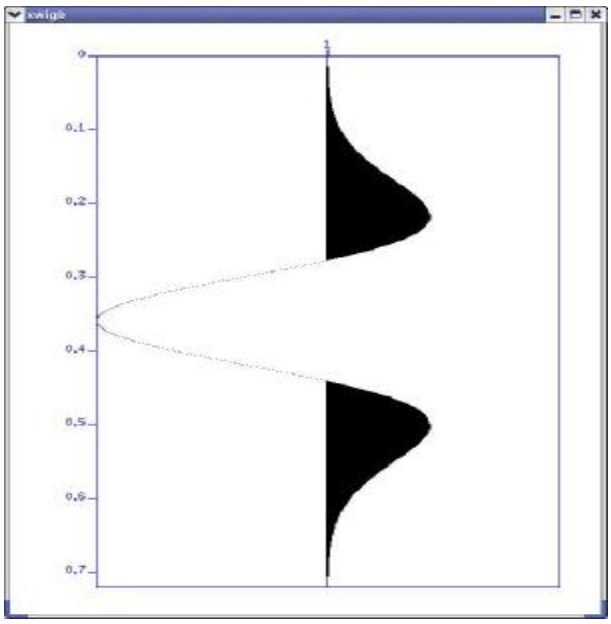
donde ns y n1 es el número de muestras, SeismicX.bin el archivo que contiene la sísmica y d1 el intervalo de muestreo de la sísmica utilizado y dt el delta de tiempo.



Visualización de la ondícula:

suaddhead < wav2.bin ns=181 | suxwigb

donde wav2.bin el archivo de la ondícula generado y ns=181 la longitud de la ondícula. Mediante este comando visualizamos la siguiente gráfica:



Usando el SEP (Stanford Exploration Project)

Para visualizar los resultados usando el SEP se debe hacer una secuencia como la que se describe a continuación:

Visualización de los Snapshots:
Se debe crear un archivo con los encabezados que se quieren desplegar, por ejemplo “Snapshot.H” cuyo contenido para el caso de los snapshots sería el siguiente:

in="SnapX.bin"
title="Snapshots en X"
n1=500
n2=500
n3=10
esize=4
d1=2
d2=2
d3=1
label1="sec"
label2="trace No."
label3="no label defined"
data_format="native_float"

Donde “in” es el nombre del archivo con los snapshots a desplegar, “n1” es el número de celdas en Z, “n2” es el número de celdas en X, “n3” es el número de Snapshots que se capturaron y “d1”, “d2” y “d3” son los respectivos deltas. Data_format es el formato en que se generan los resultados, puede ser “native_float” para maquinas como PCs con Linux o “xdr_float” para maquinas que guarden los flotantes en formato xdr.

Una vez se tiene le archivo guardado, por ejemplo “Snapshot.H”, se puede visualizar con el siguiente comando:

Grey < *Snapshot.H* |
Tube

De la misma forma se puede desplegar la sísmica y la ondícula usando los parámetros adecuados según los datos que se utilizaron para generar cada uno de estos archivos.

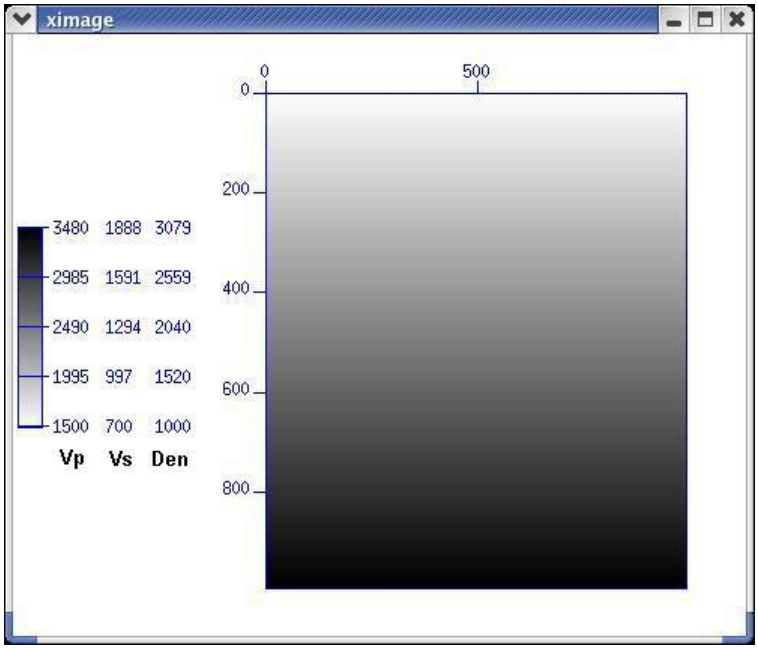
Ejemplos de Resultados

Demo1

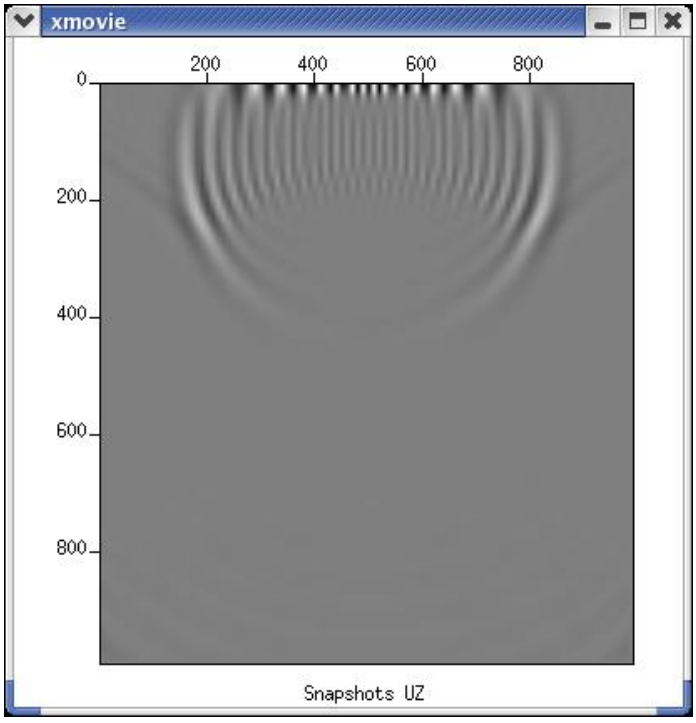
Parámetros:

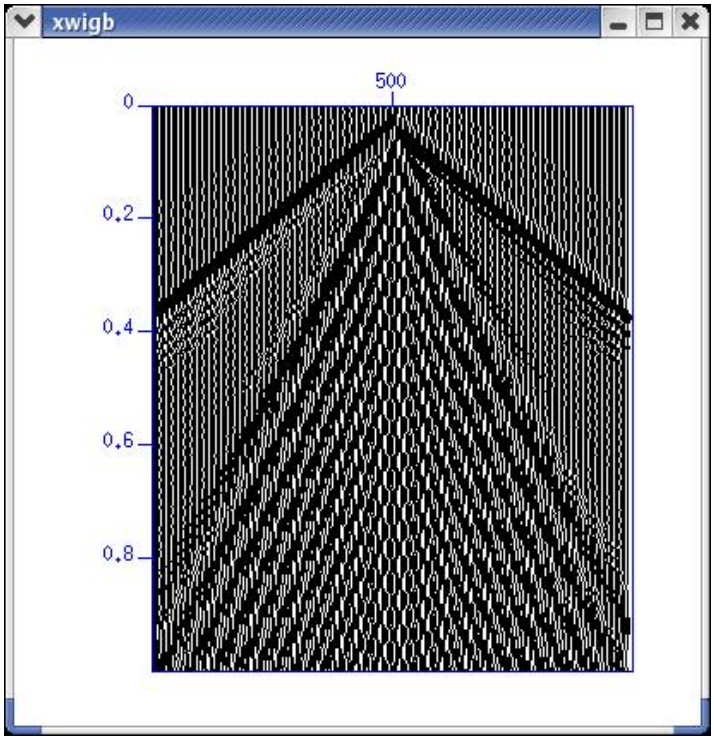
```
nx=100
nz=100
dx=10
dz=10
dt=0.0010
tmax=2
fcent=45.0
spx=500.0
spz=70.0
source_type=1  wavelet_type=3
seismograms=1 step_seismograms=0.002
snapshots=1 delay_ini=0.01 step_snapshots=0.01
higdon=1 fs=1 smooth=0
taper=0 tw=150.0 pr=25.0
vs=vs1.bin
vp=vp1.bin
den=density1.bin
pgeof=pgeof.txt
```

Modelo:


ximage
0
500
0
200
400
600
800
3480 1888 3079
2985 1591 2559
2490 1294 2040
1995 997 1520
1500 700 1000
Vp Vs Den

Resultados:


xmovie
200
400
600
800
0
200
400
600
800
Snapshots UZ



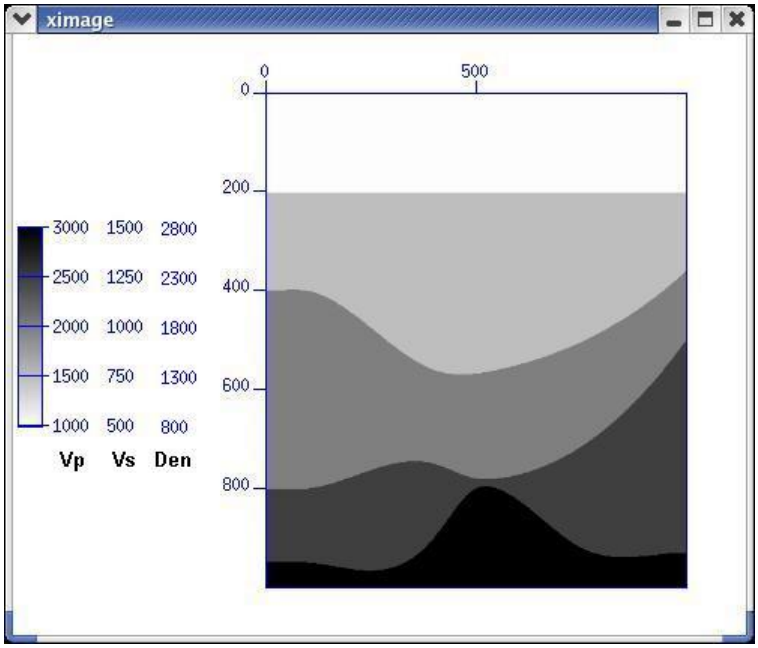
Demo2

Parámetros:

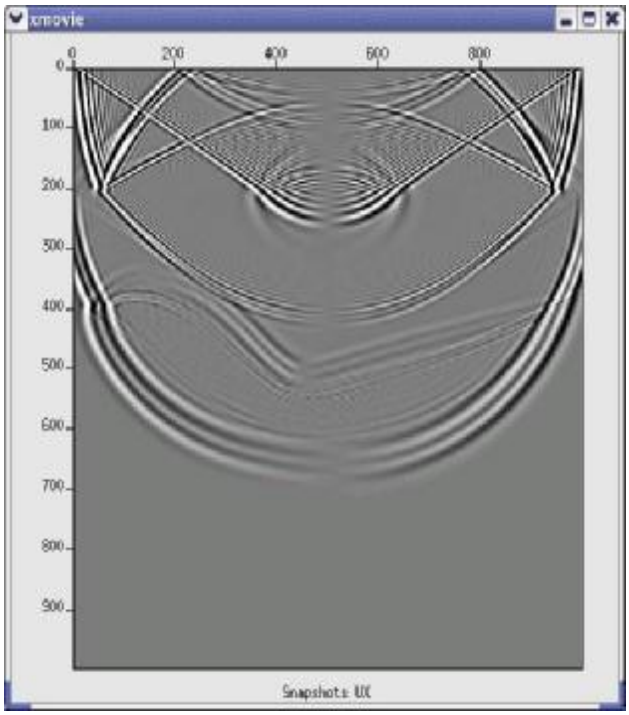
```
nx=500
nz=500
dx=2
dz=2
dt=0.0002
tmax=2
fcent=55.0
spx=500.0
spz=0.0
source_type=1  wavelet_type=3
seismograms=1 step_seismograms=0.002
snapshots=1 delay_ini=0.01 step_snapshots=0.01
higdon=1 fs=1 smooth=0
taper=0 tw=150.0 pr=25.0
vs=vs2.bin
vp=vp2.bin
den=density2.bin
pgeof=pgeof0.txt
verbose=0
file_snapx=SnapX2.bin
file_snapz=SnapZ2.bin
file_seismicx=SeismiX2.bin
file_seismicz=SeismicZ2.bin
file_snappr=SnapPR2.bin
```

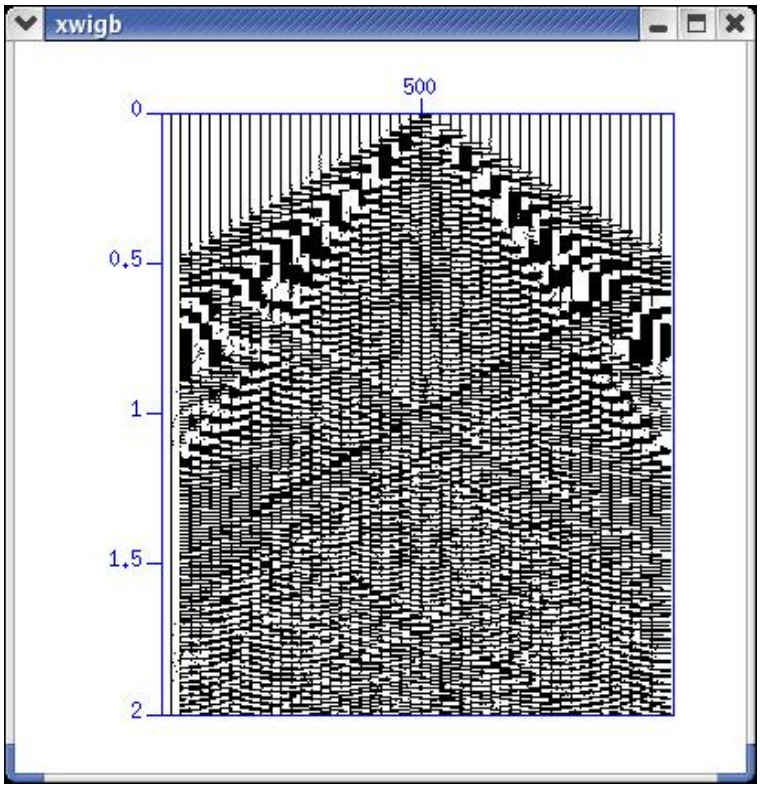
```
file_seismicpr=SeismicPR2.bin
file_out=data2.out
file_wavelet=wavelet2.bin
```

Modelo:



Resultados:





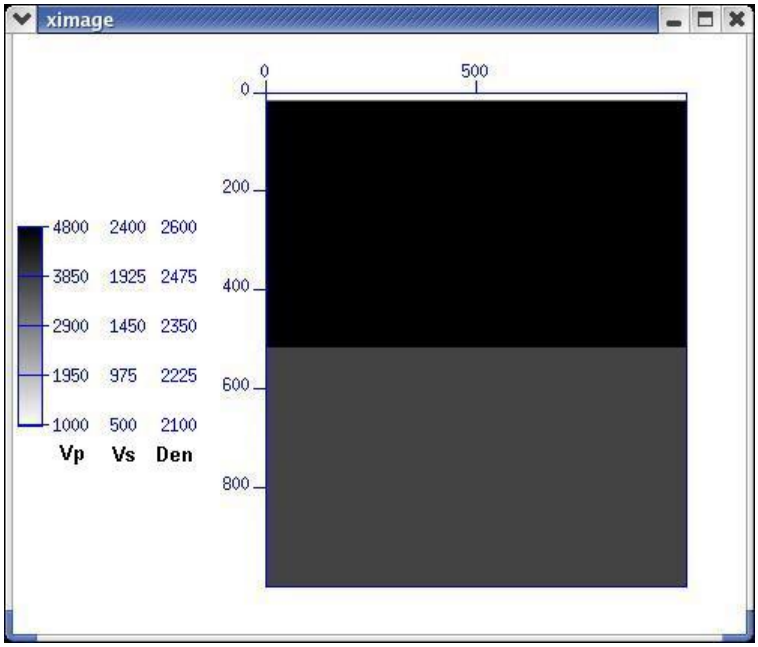
Demo3

Parámetros:

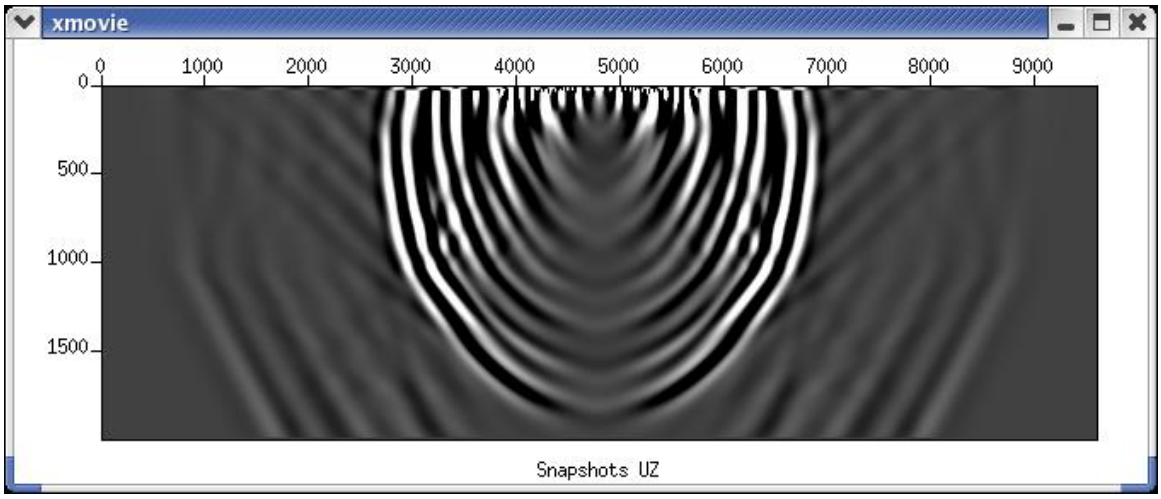
```
nx=2400
nz=500
dx=4
dz=4
dt=0.0005
tmax=3.0
fcent=9.0
spx=4800.0
spz=0.0
source_type=1  wavelet_type=3
seismograms=1 step_seismograms=0.002
snapshots=1 delay_ini=0.01 step_snapshots=0.01
higdon=1 fs=1 smooth=0
taper=0 tw=150.0 pr=25.0
vs=vs3.bin
vp=vp3.bin
den=density3.bin
pgeof=pgeof3.txt
file_snapx=SnapX3.bin
file_snapz=SnapZ3.bin
file_snappr=SnapPR3.bin
file_seismicx=SeismicX3.bin
```

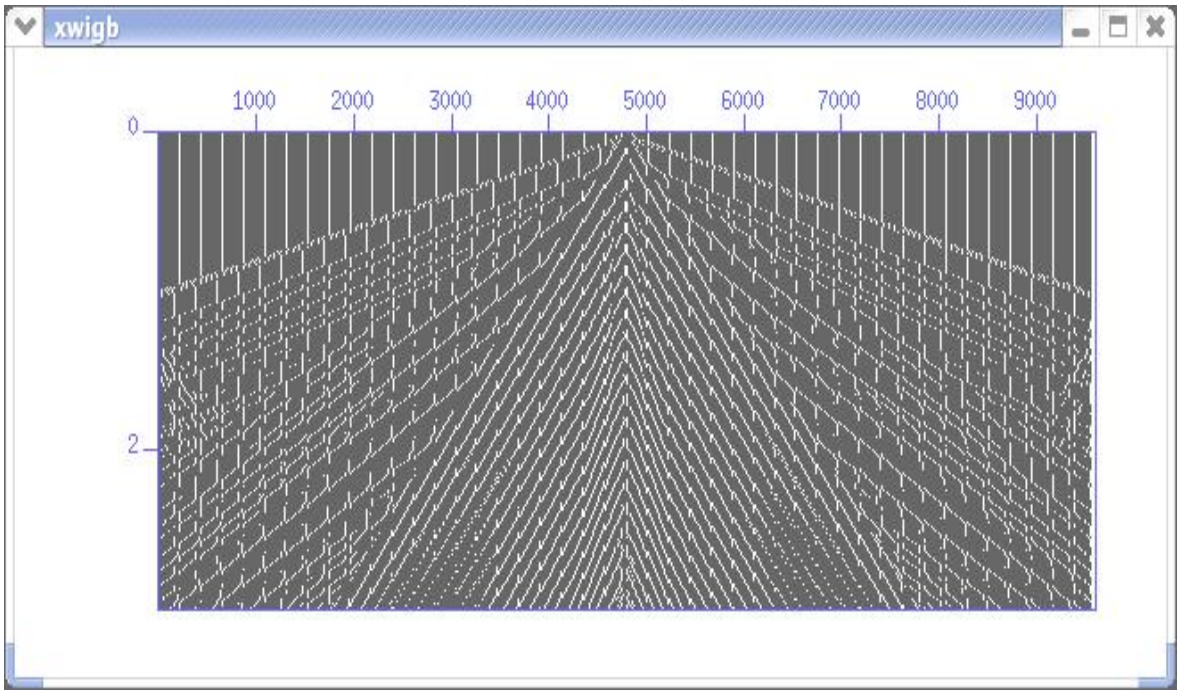
```
file_seismicz=SeismicZ3.bin
file_seismicpr=SeismicPR3.bin
file_wavelet=wavelet3.bin
file_out=data3.out
```

Modelo:



Resultados:





Créditos

Todos los derechos patrimoniales de este programa son propiedad del Instituto Colombiano del Petróleo ECOPETROL-ICP (http://www.ecopetrol.com.co).
La primera y segunda versión fue desarrollada por Numérica Ltda (http://www.numerica.com.co), Bajo la orientación técnica del grupo de geofisica de ECOPETROL – ICP, quienes desarrollaron esta Manual.
La versión paralela fue desarrollada por Diana Consuelo Hortúa Bayona, bajo el convenio de cooperación N-005 entre la Universidad Industrial de Santander (UIS) y el ICP, como trabajo de grado, y realizó la s modificiaciones necesarios para acturalizar este manual.

AUTORES:

ECOELAS2D 1.0 Y ECOELAS2D 2.0
ECOPETROL - ICP:
Tutoría:
Msc Saúl Ernesto Guevara Ochoa (sguevara@ecopetrol.com.co)
NUMERICA LTDA:
Programación:
Ing. Félix Manuel Córdoba Tuta (fcordoba@numerica.com.co)
Ing. Félix Armando Mejía Cajica (fmejia@numerica.com.co)
Asesoría Técnica:
Ing. Elkin Rafael Arroyo Negrete (earroyo@numerica.com.co)
Ing. Jairo Cesar Castañeda González (jcastago@numerica.com.co)

ECOELAS2D VERSION PARALELA

UNIVERSIDAD INDUSTRIAL DE SANTANDER

Programación

Ing. Diana Consuelo Hortúa Bayona (dianahortua@gmail.com)

Director del Proyecto

Ph.D Henry Lamos Díaz (hlamos@uis.edu.co)

Codirector del Proyecto

Ing. Víctor Martínez Abaunza ()

ECOPETROL – ICP

Tutoria:

Msc saúl Ernesto Guevara Ochoa (sguevara@ecopetrol.com.co)