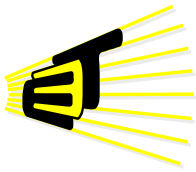
IMPLEMENTACIÓN DE UN MÓDULO DE PROPAGACIÓN DE ONDA ELÁSTICA 2D UTILIZANDO UN CLUSTER DE GPUS

ANDRÉS MAURICIO MANJARRÉS GARCÍA



Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones

Universidad Industrial de Santander

Facultad de Ingenierías Fisicomecánicas

Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones

Bucaramanga

2018

IMPLEMENTACIÓN DE UN MÓDULO DE PROPAGACIÓN DE ONDA ELÁSTICA 2D UTILIZANDO UN CLUSTER DE GPUS

Autor:
ANDRÉS MAURICIO MANJARRÉS GARCÍA

TRABAJO DE GRADO PARA OPTAR AL TÍTULO DE INGENIERO ELECTRÓNICO

Director:
Sergio Alberto Abreo Carrillo
Doctor en Ingeniería

Codirector:
Ana Beatriz Ramírez Silva
Ph.D. en Ing. Eléctrica.

Universidad Industrial de Santander
Facultad de Ingenierías Fisicomecánicas
Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones
Bucaramanga
2018

A Cecilia, por ser siempre mi motor, por impulsarme, por enseñarme cada día mas, y por ser la mejor mamá del mundo.

A Daniela, por estos años de locura, por su compañia, por tantas experiencias increibles y por siempre dar lo mejor.

A Lili, Santhiago, Lina, Daniel, Lucy y toda mi familia por ser apoyarme y estar conmigo simpre.

A todos los compañeros que estuvieron conmigo a lo largo de estos años.

AGRADECIMIENTOS

Agradezco al profesor Sergio Abreo, por su acompañamiento y disposición para ayudarme en el desarrollo de este trabajo y por siempre motivarme a ir mas lejos.

Igualmente a la profesora Ana Ramírez, por sus consejos y la confianza depositada en mi.

A Fabián Noriega, Jheyston Serrano, Johan Suarez, Katherine Flórez y Jaap Wesdorp y a todos los miembros del grupo CPS por sus aportes tan valiosos para esta investigación.

También a Paola Hernandez por su apoyo administrativo.

Este trabajo fue desarrollado con el apoyo de ECOPETROL y COLCIENCIAS como parte del proyecto de investigación No. 0266-2013.

CONTENIDO

LISTA DE FIGURAS

Pág.

LISTA DE TABLAS

RESUMEN

TÍTULO: Implementación de un módulo de propagación de onda elástica 2D utilizando un cluster de GPUs*

AUTOR: Andrés Mauricio Manjarrés García**

PALABRAS CLAVE: GPU, Ondas Elásticas, Malla Intercalada, CPML.

DESCRIPCIÓN:

Este trabajo consiste en la implementación de un algoritmo que permite modelar la propagación de ondas elásticas de dos dimensiones, en medio isótropo, es decir un medio que mantiene sus propiedades sin importar la dirección de la propagación. Adicionalmente, para evitar que la onda rebote en la frontera del modelo creando comportamientos alejados a los reales, se utilizo una región de absorción en los bordes. Esta región se desarrollo mediante la estrategia de Capas Convolucionales Perfectamente Acopladas (C-PML por sus sigas en ingles *convolutional-perfectly matched layers*).

Para implementar las ecuaciones de propagación en equipos de computo, se utilizó discretización de las derivadas por medio de diferencias finitas y de los modelos utilizando el esquema de malla intercalada. La ejecución del algoritmo se realizo sobre una unidad de procesamiento gráfico (GPU, por sus siglas en ingles) y programación en paralelo (CUDA) para acelerar el procesamiento de datos. Los resultados de la implementación numérica se contrastaron por medio de la respuesta quasi-analitica de las ecuaciones desarrollada en trabajos anteriores. Por otro lado se pudo comparar la velocidad de procesamiento del algoritmo respecto a ejecuciones que no usan computación en paralelo. El algoritmo implementado se enfoco de tal forma que el propagador pueda ser utilizado en otros procesos de análisis sísmico como el método de inversión de onda completa.

* Trabajo de grado.

** Facultad de Ingenierías Fisicomecánicas. Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones. Ingeniería Electrónica. Director: Dr. Ing. Sergio Alberto Abreo Carrillo.

ABSTRACT

TITLE: Implementation of a 2D elastic wave propagation module using a cluster of GPUs*

AUTHOR: Andrés Mauricio Manjarrés García**

KEYWORDS: Elastic Waves, GPU, Full Wave Invesion

DESCRIPTION:

This work presents the implementation of an algorithm that simulates the 2D elastic wave propagation over an isotropic medium, i.e. a medium that keeps their properties without regard the direction of propagation.In addition, to avoid non-natural reflections at the model boundaries (behavior do not expect in real propagation), this project uses an absorption zone in the borders. This zone was developed using the convolutional perfectly matched layers (C-PML) strategy.

To implement the equations on computers, it is necessary to make discretizations. This work uses finite differences to make the discretization of the derivatives and staggered-grid to discretization of the fields. The execution of the algorithm is over a Graphics Processing Unit (GPU) and parallel computing (CUDA) to accelerate the data processing. The results of the numeric implementations are contrasted with the quasi-analytic answer provided by other research. Furthermore was possible compared the processing velocity of the algorithm with regard to implementation that does not use parallels computing to demonstrate the advantages the parallel computing in propagations problems. The algorithm was focused on a goal to use the propagator in seismic data processing like the full-wave inversion method.

* Trabajo de grado.

** Facultad de Ingenierías Fisicomecánicas. Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones. Ingeniería Electrónica. Director: Dr. Ing. Sergio Alberto Abreo Carrillo.

INTRODUCCIÓN

En la exploración geofísica se busca determinar las características de los materiales presentes en el subsuelo, para esto se utilizan métodos de exploración como los electromagnéticos o los sísmicos. En el caso de exploración sísmica, los datos obtenidos en campo, son procesados para obtener imágenes sísmicas ya sea de velocidad, de densidad o de otras propiedades que permitan conocer los materiales. En este procesamiento de datos sísmicos existen herramientas para acercar la información inferida de los datos un poco más a la realidad. Una de estas herramientas es la Inversión de Onda Completa (FWI por sus siglas en inglés Full Waveform Inversion) que es un procedimiento que toma como punto de partida un primer modelo y simula el comportamiento de la onda replicando el experimento de campo para obtener resultados que luego son comparados con datos reales. A partir de estas comparaciones se mejora el modelo inicial.

Para poder realizar el proceso de inversión de onda completa es necesario modelar la interacción entre las ondas y el medio. En esta interacción se generan varios tipos de ondas, que se pueden clasificar en ondas volumétricas y en ondas superficiales. Las ondas superficiales no serán de estudio en este trabajo ya que son ondas de poca penetración, no tan útiles para procesos de obtención de imágenes del subsuelo. Para simular las ondas volumétricas se utilizan un conjunto de ecuaciones derivadas de la ley de Hook para esfuerzos pequeños, las cuales simulan la propagación de la onda sobre un medio isótropo, es decir un medio en el que sus propiedades físicas son independientes de la dirección [1].

La propagación es expresada en ecuaciones diferenciales, las cuales se discretizan utilizando diferencias finitas y el esquema de malla intercalada descrito por Yee [2], el cual es aplicado al caso de ondas elásticas por Virieux en [3]. Además, es necesario aplicar términos adicionales debido al problema de condiciones de frontera, solucionado con la técnica Condición de contorno absorbente de capa convolucional perfectamente adaptada [4] (CPML por sus siglas en inglés(Convolutional Perfectly Matched Layer). Debido a que la mayoría de operaciones se realizan sobre datos en forma matricial, el algoritmo es altamente paralelizable, por lo que se decidió implementar las ecuaciones en una Unidad de Procesamiento Gráfico (GPU) para reducir los tiempos de ejecución en comparación a implementaciones en CPU.

Este documento contiene el siguiente orden. En el capítulo 2 se explica la propagación de las ondas en medios elásticos y las ecuaciones que modelan este fenómeno. El capítulo 3 brinda la información relevante a la implementación de los algoritmos. En el capítulo 4 se describen las pruebas realizadas y los resultados ob-

tenidos y finalmente el quinto capítulo ofrece las conclusiones finales y sugerencias para trabajos futuros.

1. OBJETIVOS

1.1. OBJETIVO GENERAL

Implementar un algoritmo que modele la propagación de una onda en un medio elástico sobre un cluster de GPUs, para ser utilizado en posteriores procesos de inversión de onda completa (FWI).

1.2. OBJETIVOS ESPECÍFICOS

1. Implementar un algoritmo de propagación de onda elástica incluyendo condiciones de frontera absorbente para evitar los rebotes no naturales en una GPU.
2. Generar un reporte de los recursos utilizados por la GPU.
3. Adecuar y almacenar los datos necesarios para un posterior uso del propagador en una FWI elástica.

2. PROPAGACIÓN DE ONDAS ELÁSTICAS

2.1. PRINCIPIOS BÁSICOS DE LA PROPAGACIÓN DE ONDAS ELÁSTICAS

Las ondas elásticas se propagan siguiendo dos ecuaciones fundamentales. La ecuación 2.1, llamada ecuación de Cauchy para la conservación de la cantidad de movimiento, sin tener en cuenta las fuerzas que no implican contacto físico como la gravedad o campos eléctricos. Y la segunda ecuación es la ley de Hooke 2.2 con la que podemos relacionar los esfuerzos con las deformaciones ε para el medio elástico

$$\rho \frac{\partial v_i}{\partial t} = \sum_{j=1}^{3} \frac{\partial \tau_{ij}}{\partial x_j}, i = \{1, 2, 3, \} \tag{2.1}$$

En la ecuación de Cauchy v_i representa la velocidad de propagación, ρ la densidad del medio y τ los esfuerzos

$$\tau_{ij} = \sum_{l=1}^{3} \sum_{k=1}^{3} C_{ijkl} \cdot \varepsilon_{kl} \qquad i, j = \{1, 2, 3\}, \tag{2.2}$$

donde C_{ijkl} representa el tensor de elasticidad el cual está compuesto por 81 constantes que dependen del material que compone el medio y ε_{kl} representa el tensor de deformaciones. Debido a cierta simetría que tiene el subsuelo en los coeficientes de las constantes elásticas, el tensor C_{ijkl} se reduce de 81 a 36 constantes y cambiando la notación se reducen aún más a solo 21 constantes. Para este trabajo de grado se decidió trabajar con un medio isótropo el cual es un medio que mantiene sus propiedades sin importar la dirección de las fuerzas. Esto hace que se reduzcan las constantes de elasticidad a solo dos constantes independientes, llamadas módulos de elasticidad de Lamé (μ y λ). μ representa la rigidez del medio y λ tiene una definición física un poco más compleja que depende de otros paráme-

tros pero que en general representa la compresibilidad del medio. Finalmente con solo estas constantes obtenemos la siguiente matriz de coeficientes

$$C_{Isotropo} = \begin{pmatrix} \lambda+2\mu & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda+2\mu & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & \lambda+2\mu & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{pmatrix}. \tag{2.3}$$

Adicionalmente, el tensor ε_{kl} de deformaciones puede ser descrito por desplazamientos infinitesimales como,

$$\varepsilon_{kl} = \frac{1}{2}\left(\frac{\partial u_k}{\partial x_l} + \frac{\partial u_l}{\partial x_k}\right), \qquad k,l = \{1,2,3\}, \tag{2.4}$$

donde u_k y u_l representan estos desplazamientos. Por último, teniendo en cuenta las ecuaciones 2.1, 2.2 y 2.4 y eliminando todos los componentes de la dirección y debido a que el trabajo se enfoca en solo dos dimensiones, se construye el sistema de ecuaciones 2.5 que modelan la propagación 2D de ondas elásticas sobre un medio isótropo

$$\begin{aligned} \frac{\partial v_x}{\partial t} &= \frac{1}{\rho}\left(\frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{xz}}{\partial x}\right) \\ \frac{\partial v_z}{\partial t} &= \frac{1}{\rho}\left(\frac{\partial \tau_{zx}}{\partial z} + \frac{\partial \tau_{zz}}{\partial z}\right) \\ \frac{\partial \tau_{xx}}{\partial t} &= (\lambda+2\mu)\frac{\partial v_x}{\partial x} + \lambda\frac{\partial v_z}{\partial z} + f\,(t) \\ \frac{\partial \tau_{zz}}{\partial t} &= \lambda\frac{\partial v_x}{\partial x} + (\lambda+2\mu)\,\frac{\partial v_z}{\partial z} + f\,(t) \\ \frac{\partial \tau_{xz}}{\partial t} &= \mu\frac{\partial v_x}{\partial z} + \mu\frac{\partial v_z}{\partial x} \end{aligned} \tag{2.5}$$

donde $f\,(t)$ representa la fuente de excitación externa que simula una explosión. Agregada solamente en los tensores de esfuerzos en direcciones horizontales τ_{xx} y verticales τ_{zz} [5]. La fuente utilizada es conocida como ondícula Ricker y se define como

$$f\,(t) = (1 - 2\pi^2 f_q^2 (t-t_0)^2) e^{-\pi^2 f_q^2 (t-t_0)^2} \tag{2.6}$$

donde f_q representa la frecuencia central de la fuente, y t_0 es el desplazamiento en tiempo de la fuente, normalmente definido como $1/f_q$.

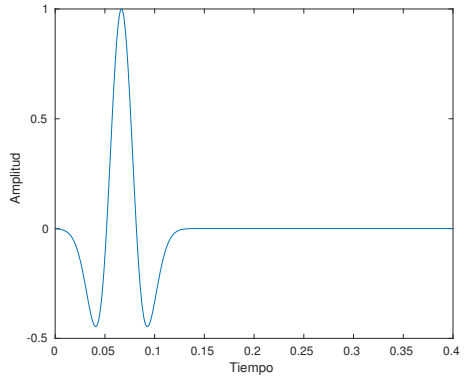


Figura 1: Ondícula Ricker con frecuencia $f_q = 15Hz$, $t_0 = 1/f_q$ y un tiempo total $t = 0{,}4s$

2.2. CONDICIONES DE FRONTERA ABSORBENTES

En las pruebas de campo las ondas sísmicas se propagan libremente en el medio. Al momento de implementar las ecuaciones de propagación, las fronteras son limitadas debido a la memoria del equipo, ocasionando que las ondas reboten en las fronteras y generen reflexiones que no son propias del comportamiento real de la onda. Para solucionar esto se utilizó la estrategia de capas convolucionales perfectamente acopladas (C-PML, por sus siglas en inglés convolutional-perfectly matched layers)[4] la cual disminuye las reflexiones introduciendo un término dependiente de la frecuencia que actuará como filtro para las ondas que se reflejan. La implementación consiste en cambiar el operador de derivada que se encuentran en la ecuación 2.5 por el operador 2.7.

$$\bar{\partial} \approx \partial_s + \Psi_s, \qquad s = \{x, z\} \tag{2.7}$$

donde Ψ_s representa el amortiguamiento, y necesita de su valor en el tiempo anterior para ser calculado como se indica en la siguiente expresión

$$\Psi_s = b_s \Psi_s^{n-1} + a_s(\partial_s), \tag{2.8}$$

donde a_s y b_s representan los coeficientes de amortiguamiento definidos por las ecuaciones:

$$\begin{aligned} a_s &= \frac{d_s}{d_s + \alpha_s}\,(b_s - 1) \\ b_s &= e^{-(d_s+\alpha_s)\Delta t} \end{aligned} \tag{2.9}$$

Los parámetros se definieron según [6]; Donde α es una variable que disminuye linealmente a medida que se aleja del límite del modelo. El valor inicial de esta variable es $\alpha_{max} = \pi f_0$ siendo f_0 la frecuencia de la fuente. $d_s = d_0 \left(\frac{X}{L}\right)^N$, donde X es la distancia en metros dentro de la región CPML en dirección s, N es el orden de amortiguamiento, que usualmente es 2, L es la distancia total de la zona CPML en metros y d_0 está dada por $d_0 = -(N+1)\, v_p \frac{(R_c)}{2L}$, donde R_c fue tomado como el coeficiente de reflexión teórico $R_c = 0{,}001$ [7].

La ecuación 2.5 con el cambio de operador se muestra a continuación, resaltando que Ω y Ψ se determinan con la expresión 2.8, pero Ω es para los esfuerzos y Ψ para las velocidades.

$$\begin{aligned}
\frac{\partial v_x}{\partial t} &= \frac{1}{\rho}\left(\frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{xz}}{\partial x}\right) - \frac{1}{\rho}\left(\Omega_{xx_x} + \Omega_{xz_z}\right) \\
\frac{\partial v_z}{\partial t} &= \frac{1}{\rho}\left(\frac{\partial \tau_{zx}}{\partial z} + \frac{\partial \tau_{zz}}{\partial z}\right) - \frac{1}{\rho}\left(\Omega_{xz_x} + \Omega_{zz_z}\right) \\
\frac{\partial \tau_{xx}}{\partial t} &= (\lambda + 2\mu)\frac{\partial v_x}{\partial x} + \lambda \frac{\partial v_z}{\partial z} - (\lambda + 2\mu)\Psi_{xx} - \lambda \Psi_{zz} + f\,(t) \\
\frac{\partial \tau_{zz}}{\partial t} &= \lambda \frac{\partial v_x}{\partial x} + (\lambda + 2\mu)\,\frac{\partial v_z}{\partial z} - (\lambda + 2\mu)\Psi_{zz} - \lambda \Psi_{xx} + f\,(t) \\
\frac{\partial \tau_{xz}}{\partial t} &= \mu \frac{\partial v_x}{\partial z} + \mu \frac{\partial v_z}{\partial x} - \mu \Psi_{zx} - \mu \Psi_{xz}
\end{aligned} \tag{2.10}$$

2.3. DISCRETIZACIÓN Y USO DE MALLA INTERCALADA

El uso de ecuaciones matemáticas que deben ser implementadas en equipos de cómputo hace necesario la discretización para aproximar operadores mediante operaciones matemáticas más simples. En este trabajo las derivadas se aproximaron mediante el uso de diferencias finitas de primer orden centradas como se muestra en la ecuación 2.11.

$$\frac{\partial f\,(x)}{\partial x} = \frac{f\,(x + 1/2 \cdot \Delta h) - f\,(x - 1/2 \cdot \Delta h)}{\Delta h} \tag{2.11}$$

donde Δh representa el salto espacial. Adicionalmente las ecuaciones 2.10 que describen la propagación poseen una dependencia temporal; por ejemplo si queremos calcular el valor de ∂T_{xx} en el tiempo $t = 1s$ necesitamos conocer el valor de V_x

en ese mismo tiempo, el cual aún no está definido. Esto se soluciona utilizando un campo de tiempo intermedio y utilizando la técnica llamada malla intercalada [2].

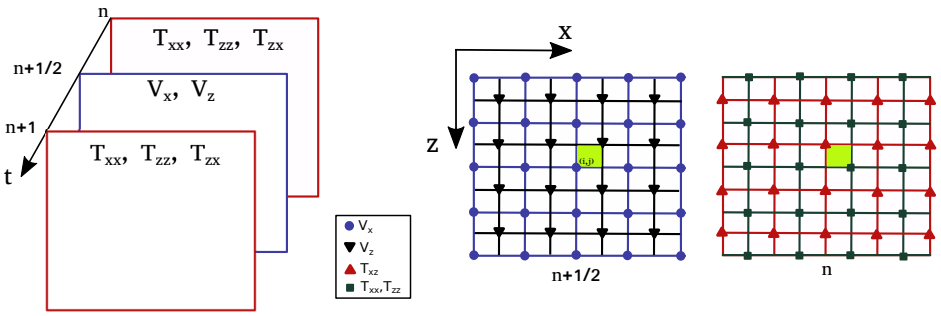


Figura 2: Esquema De Malla Intercalada: La malla azul representa el campo de velocidad en X, la malla negra el campo de velocidad en Z, la malla roja el tensor de esfuerzo en XZ y la malla verde los tensores de esfuerzo en XX y ZZ. La distancia entre los puntos de cada malla se define como Δh. Adicionalmente las mallas de velocidad se definen en un tiempo intermedio $n + \frac{1}{2}$ donde n es el numero de la iteración de tiempo.

En la figura 2 se puede observar cómo se realizan los desplazamientos en tiempo y en espacio, y haciendo uso de diferencias finitas podemos obtener las siguientes

ecuaciones discretizadas incluyendo la región de absorción:

$$
\begin{aligned}
V^{n+\frac{1}{2}}_{x(i,j)} &= V^{n-\frac{1}{2}}_{x(i,j)} + \frac{\Delta t}{\rho\,(i,j)}\left[D_x T^n_{xx(i,j)} - \Omega^{n+\frac{1}{2}}_{xx_x(i,j)} + D_z T^n_{xz(i,j)} - \Omega^{n+\frac{1}{2}}_{xz_z(i,j)}\right] \\
V^{n+\frac{1}{2}}_{z\left(i+\frac{1}{2},j+\frac{1}{2}\right)} &= V^{n-\frac{1}{2}}_{z\left(i+\frac{1}{2},j+\frac{1}{2}\right)} + \frac{\Delta t}{\rho\left(i+\frac{1}{2},j+\frac{1}{2}\right)}\left[D_x T^n_{xz\left(i+\frac{1}{2},j+\frac{1}{2}\right)} - \Omega^{n+\frac{1}{2}}_{xz_x\left(i+\frac{1}{2},j+\frac{1}{2}\right)}\right] \\
&\quad + \frac{\Delta t}{\rho\left(i+\frac{1}{2},j+\frac{1}{2}\right)}\left[D_z T^n_{zz\left(i+\frac{1}{2},j+\frac{1}{2}\right)} - \Omega^{n+\frac{1}{2}}_{zz_z\left(i+\frac{1}{2},j+\frac{1}{2}\right)}\right] \\
T^{n+1}_{xx\left(i+\frac{1}{2},j\right)} &= T^n_{xx\left(i+\frac{1}{2},j\right)} + \Delta t\left[\left(\lambda_{\left(i+\frac{1}{2},j\right)} + 2\mu_{\left(i+\frac{1}{2},j\right)}\right)\left(D_x V^{n+\frac{1}{2}}_{x\left(i+\frac{1}{2},j\right)} - \Psi^{n+1}_{xx\left(i+\frac{1}{2},j\right)}\right)\right] \\
&\quad + \Delta t\left[\lambda_{\left(i+\frac{1}{2},j\right)}\left(D_z V^{n+\frac{1}{2}}_{z\left(i+\frac{1}{2},j\right)} - \Psi^{n+1}_{zz\left(i+\frac{1}{2},j\right)}\right)\right] \\
T^{n+1}_{zz\left(i+\frac{1}{2},j\right)} &= T^n_{zz\left(i+\frac{1}{2},j\right)} + \Delta t\left[\left(\lambda_{\left(i+\frac{1}{2},j\right)} + 2\mu_{\left(i+\frac{1}{2},j\right)}\right)\left(D_z V^{n+\frac{1}{2}}_{z\left(i+\frac{1}{2},j\right)} - \Psi^{n+1}_{zz\left(i+\frac{1}{2},j\right)}\right)\right] \\
&\quad + \Delta t\left[\lambda_{\left(i+\frac{1}{2},j\right)}\left(D_x V^{n+\frac{1}{2}}_{x\left(i+\frac{1}{2},j\right)} - \Psi^{n+1}_{xx\left(i+\frac{1}{2},j\right)}\right)\right] \\
T^{n+1}_{xz\left(i,j+\frac{1}{2}\right)} &= T^n_{zz\left(i,j+\frac{1}{2}\right)} + \Delta t\mu_{\left(i,j+\frac{1}{2}\right)}\left[D_z V^{n+\frac{1}{2}}_{x\left(i,j+\frac{1}{2}\right)} + D_x V^{n+\frac{1}{2}}_{z\left(i,j+\frac{1}{2}\right)}\right] \\
&\quad - \Delta t\mu_{\left(i,j+\frac{1}{2}\right)}\left[\Psi^{n+1}_{xz\left(i,j+\frac{1}{2}\right)} + \Psi^{n+1}_{zx\left(i,j+\frac{1}{2}\right)}\right]
\end{aligned}
\tag{2.12}
$$

donde Δt representa el tamaño del paso de tiempo determinado para la discretización. Este valor junto con el Δh deben ser calculados teniendo en cuenta la ecuación de estabilidad para esquemas en diferencias finitas 2.13 definido en [8].

$$
C = v_p \Delta t \sqrt{\frac{1}{\Delta_x^2} + \frac{1}{\Delta_z^2}} < 1 \tag{2.13}
$$

En esta expresión v_p representa la velocidad de la onda p, la cual es un dato de entrada para el algoritmo y $\Delta x = \Delta z = \Delta h$ en nuestro caso. Por último es necesario tener en cuenta que para modelos no homogéneos, es decir que sus parámetros cambian dependiendo de la posición, es necesario interpolar los parámetros utilizando las ecuaciones 2.14.

$$
\begin{aligned}
S_{\left(i+\frac{1}{2},j\right)} &= \frac{1}{2}\left[S_{(i,j)} + S_{(i+1,j)}\right] \\
S_{\left(i,j+\frac{1}{2}\right)} &= \frac{1}{2}\left[S_{(i,j)} + S_{(i,j+1)}\right] \\
S_{\left(i+\frac{1}{2},j+\frac{1}{2}\right)} &= \frac{1}{4}\left[S_{(i,j)} + S_{(i,j+1)} + S_{(i+1,j+1)} + S_{(i+1,j)}\right]
\end{aligned}
\tag{2.14}
$$

donde S puede representar cualquiera de los parámetros μ, ρ o λ.

3. IMPLEMENTACIÓN DE ALGORITMO EN GPU

El algoritmo se implementó como lo indica el siguiente diagrama de flujo,

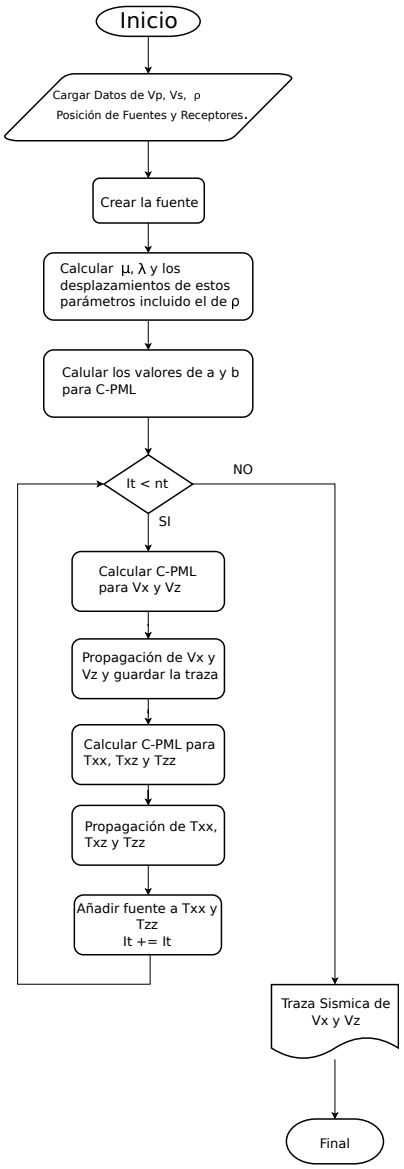


Figura 3: Algoritmo de propagación elástica implementado

en donde los datos de entrada y salida son archivos binarios. La fuente se creó utilizando la ecuación 2.6 y los parámetros λ y μ, se calculan utilizado las siguientes ecuaciones,

$$v_s = \sqrt{\frac{\mu}{\rho}}, \quad v_p = \sqrt{\frac{\lambda + 2\mu}{\rho}}, \tag{3.1}$$

con sus desplazamientos utilizando las ecuaciones 2.14. Las constantes a y b definidas en la ecuación 2.9, son iguales en todo el tiempo de la propagación por lo que solo se definen una vez. Posterior a la definición de las constantes de C-PML las iteraciones de tiempo comienzan con It=0 y donde $nt = \frac{Tiempo\_de\_propagacion}{\Delta t}$. Debido a la necesidad de usar malla intercalada (Figura 2), primero se calculan las velocidades y luego los esfuerzos. Antes de cada cálculo de velocidad o esfuerzo se determinan los valores de las variables de C-PML Ω y Ψ por aparte para facilitar la programación. En la última parte de cada iteración se añade la fuente (Figura 1) en los campos τ_{xx} y τ_{zz}. Finalmente al terminar las iteraciones se guardan dos archivos binarios que cuentan con los datos de los receptores. Un archivo contiene solo información del campo V_x y el otro archivo solo información del campo V_z. Solo se capturan estos dos datos, ya que en campo los receptores utilizados con frecuencia, son geófonos [9], los cuales pueden capturar solo esta información.

3.1. COMPUTACIÓN EN PARALELO

La computación en paralelo es una disciplina en la que muchos cálculos son ejecutados en el mismo instante de tiempo. Desde el inicio del desarrollo de la computación se ha buscado paralelizar procesos haciendo que procesadores utilicen memoria compartida y puedan trabajar juntos en un mismo conjunto de datos.

En la actualidad uno de los dispositivos que trabaja este tipo de computación son las unidades de procesamiento gráfico (GPU por sus siglas en inglés). Las GPUs son dispositivos que cuentan en su interior con una arquitectura computacional dedicada para procesamiento de datos en paralelo. En el caso de NVIDIA, empresa que fabrica GPUs, esta arquitectura es conocida como CUDA. CUDA esta basada en un conjunto de núcleos de ejecución denominadas *Streaming Multiprocessors (SMs)*, cada uno de estos SM está a su vez compuesto por “Núcleos” CUDA los cuales están encargados de ejecutar las instrucciones requeridas. NVIDA con CUDA introdujo también un entorno de programación basado en los estándares C/C++ llamado **CUDA C/C++**. Este lenguaje extiende las funciones de C, para que el programador pueda utilizar un nuevo tipo de función llamado *kernel*. Los *kernel* son funciones ejecutadas N veces por N *threads*, donde N es un número natural que depende de la capacidad de la GPU. Cuando se ejecuta un *kernel* una parte de la GPU se encarga de determinar que *Streaming Multiprocessors* está libre y utilizar sus núcleos CUDA para implementar todos los *threads*. En la definición del *kernel*

el programador debe definir cuantos *threads* se desean implementar utilizando dos parámetros, tamaño del bloque y tamaño de la malla.

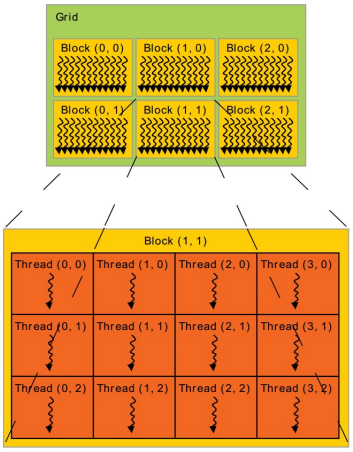


Figura 4: Malla de bloques en configuración de dos dimensiones, dentro de la GPU: Los cuadrados amarillos representan los bloques y dentro de ellos, las flechas curvas son los *threads*. En esta imagen se observa una malla con 6 bloques acomodados en dos filas de tres columnas cada una. Dentro de cada bloque existen 12 *threads* en 3 filas de 4 columnas cada una. Tomado de [10]

En la figura 4 se puede ver que una malla(*Grid* en la imagen) está formada por un número definido de bloques los cuales a su vez también están conformados por un número de *threads*. La malla y los bloques se pueden definir hasta en tres dimensiones. Una de las aplicaciones a nivel de investigación de la computación en paralelo son las operaciones sobre datos en forma matricial. El algoritmo 3.1 muestra un ejemplo del uso de esta programación.

```
1  #include <stdio.h>
   #include <stdlib.h>
3  #include <string.h>


5  __global__ void suma(float *A, float *B, float *C){
     int i = threadIdx.x + blockDim.x * blockIdx.x;
7

     C[i] = A[i]+B[i];
9  }


11

   int main()
```

```
13 {
       ...
15
       suma<<<3,5>>>(A, B, C);
17
       ...
19 }
```

Algoritmo 3.1: Algoritmo ejemplo para calcular una suma de matrices

Este es un ejemplo de un *kernel* que realiza una suma entre dos vectores. Los vectores poseen 15 elementos por lo que se ejecutan 15 *threads*. Estos a su vez se dividen en 3 bloques, donde cada bloque posee 5 *threads*. Esto está especificado por los números dentro de los «`Bloques_en_malla,Threads_en_bloque` ». El compilador de NVDIA al ser una extensión de código de C puede compilar código paralelo y código serie, el código en serie se ejecuta en una CPU y el paralelo (aquel que se define como función ___global___) sobre una GPU, en términos de CUDA C la CPU se conoce *Host* y la GPU como *Device*

3.2. EQUIPO UTILIZADO

Para la implementación de este proyecto se utilizó un nodo del clúster GPU, perteneciente al grupo de investigación CPS de la Universidad Industria de Santander. El nodo esta conformado por,

Recursos de **CPU**: Dos procesadores Intel® Xeon® Processor E5-2670 v3, con las siguientes características cada uno :

- 12 Núcleos.
- 24 Hilos.
- Frecuencia Base 2,30 GHz.

Recursos de **GPU**: Dos GPUs Tesla k40 de NVIDIA, con las siguientes características cada uno :

- 15 *Streaming Multiprocessors*.
- 192 núcleos cuda.

- Máximo de 2048 *threads* por SM.

De forma general el clúster cuenta con :

- RAM : 192 GB.
- Disco duro de 10 TB.
- Conexión Ethernet de 10 Mb.

El siguiente esquema muestra una representación gráfica de un nodo del clúster para ayudar en la compresión, este esquema no contiene las especificaciones técnicas, y tampoco representa a la perfección las conexiones internas del clúster.

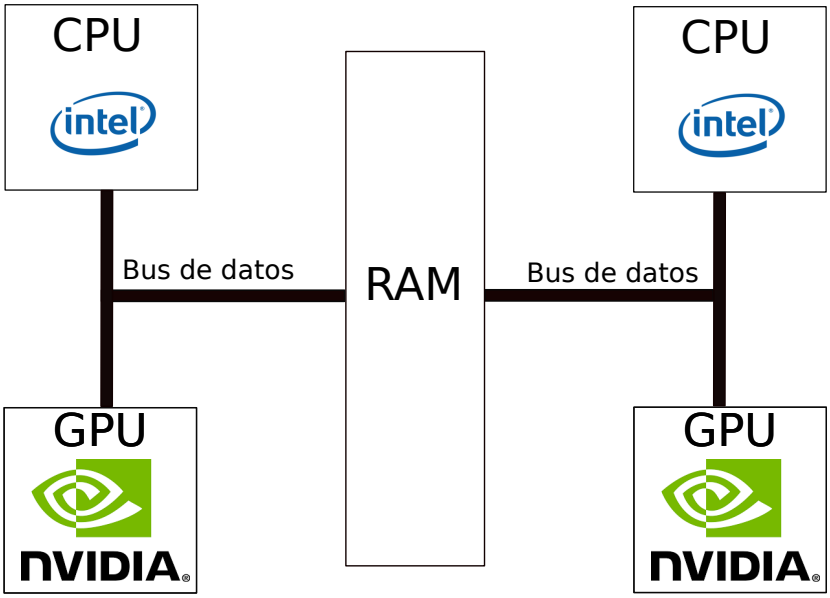


Figura 5: Representación gráfica de un nodo del clúster utilizado para la implementación del algoritmo

4. RESULTADOS

4.1. ABSORCIÓN EN EL BORDE

Para comprobar que los bordes estuvieran absorbiendo la energía, se propagó sobre un modelo constante de 3000[m] de largo y 3000[m] de profundidad, con $v_p = 2500[m/s]$, $v_s = 1558[m/s]$ y $\rho = 1500[Kg/m^3]$. La energía para ondas elásticas se mide como combinación de la energía cinética K dada por el movimiento de las partículas y de la energía potencial elástica T almacenada por los desplazamientos de estas partículas de su posición de equilibrio. La expresión para determinar estas energías está dada por

$$E = K + T = \left[\frac{1}{2}\rho\left(v_x^2 + v_z^2\right)\right] + \left[\frac{1}{2}\left(\tau_{xx}\cdot\varepsilon_{xx} + \tau_{zz}\cdot\varepsilon_{zz} + 2\tau_{xz}\cdot\varepsilon_{xz}\right)\right], \tag{4.1}$$

donde

$$\begin{aligned}
\varepsilon_{xx} &= (\lambda + 2\mu)\tau_{xx} - \frac{\lambda\tau_{zz}}{4\mu(\lambda+\mu)} \\
\varepsilon_{zz} &= (\lambda + 2\mu)\tau_{zz} - \frac{\lambda\tau_{xx}}{4\mu(\lambda+\mu)} \\
\varepsilon_{xz} &= \frac{\tau_{xz}}{2\mu}.
\end{aligned} \tag{4.2}$$

Finalmente los resultados obtenidos se pueden ver en la Figura 6, donde se comprueba que la energía de la propagación disminuye para el caso donde se implementan las ecuaciones de frontera absorbente.

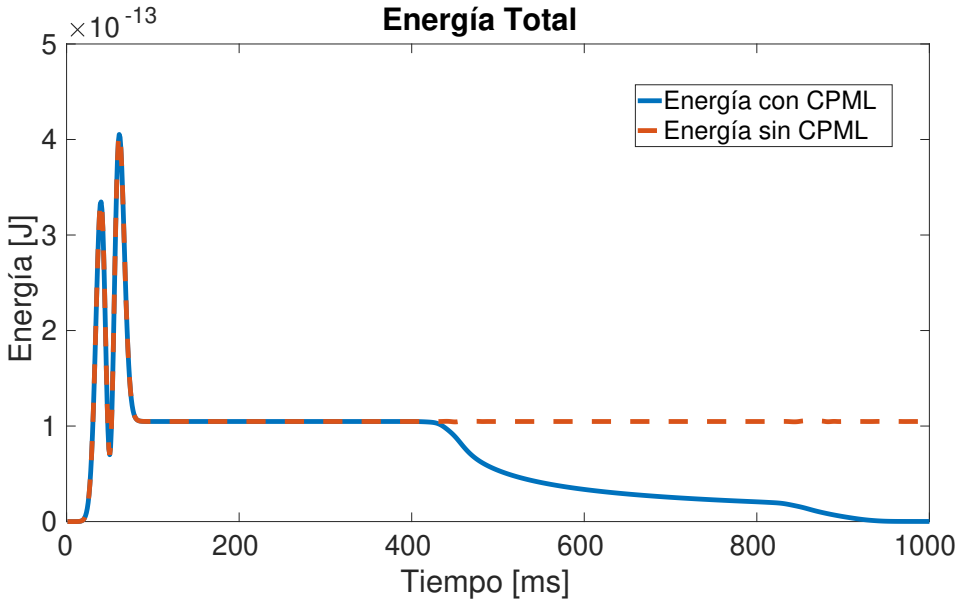


Figura 6: Energía de la propagación con y sin C-PML: La línea punteada representa la energía cuando no hay región de absorción por lo se puede observar que la energía se queda atrapada, presentando un comportamiento no natural. Por otro lado línea sólida de color azul, permite observar que la energía cuando se implementó la región de absorción es disipada, comportamiento esperado en el experimento de campo que se desea simular.

En esta imagen también se puede observar que los primeros 400 [ms] para ambos casos(con y sin C-PML) son iguales. Antes de los 400[ms] la onda aun no ingresa a la zona de absorción por lo que se espera que la energía en hasta este instante de tiempo sea la misma.

4.2. COMPROBACIÓN DE LA IMPLEMENTACIÓN NUMÉRICA

Para validar la implementación de las ecuaciones, se propago sobre el modelo descrito en la Figura 7. El dato capturado por el receptor se comparo con un dato quasi-analitico generado por el software EX2DELEL ofrecido por el proyecto Spice (*Seismic wave Propagation and Imaging in Complex media: a European network* [11]). Se realizaron dos comparaciones, la primera con un tiempo de propagación de 1 segundo y la segunda con un tiempo de 1,5 segundos.

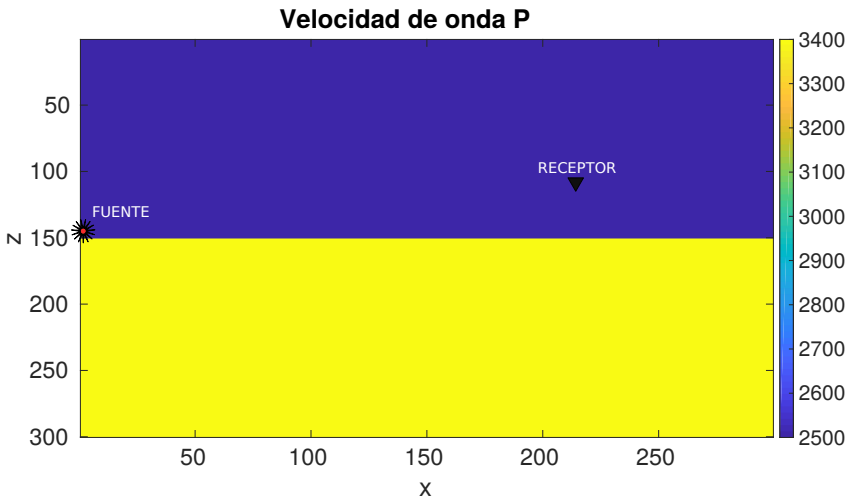
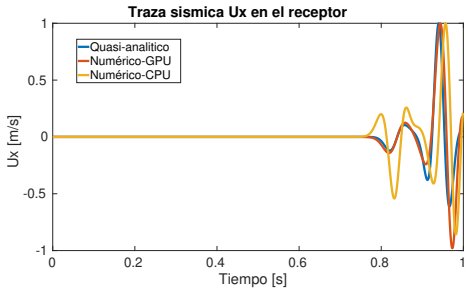
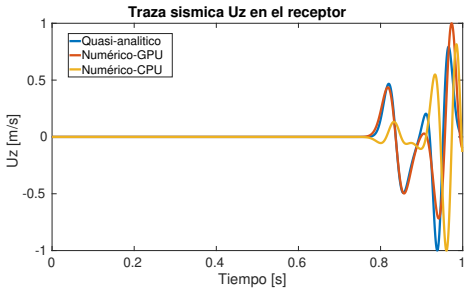


Figura 7: La fuente se encuentra a 10[m] en X y 1140[m] en Z. El receptor está a 2140 [m] en X y 1090[m] en Z, Δh para el modelo fue de 10[m]. La parte superior del modelo(Azul) tiene una velocidad de la onda p $v_p = 2500[m/s]$; velocidad de onda s $v_s = 1558[m/s]$ y densidad $\rho = 1500[Kg/m^3]$. La parte inferior (Amarilla) tiene una velocidad de la onda p $v_p = 3400[m/s]$; velocidad de onda s $v_s = 1963{,}0484[m/s]$ y densidad $\rho = 2600[Kg/m^3]$. Valores por defecto en software EX2DELEL.

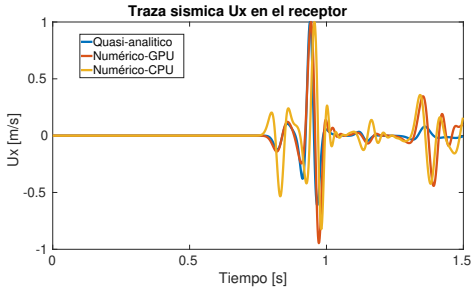
Con el objetivo de comparar con la implementación en CPU, se realizó la misma prueba para un código que realiza la misma implementación del algoritmo de las ecuaciones 2.12, pero sobre una de las CPU del cluster. Obteniendo los siguientes resultados:



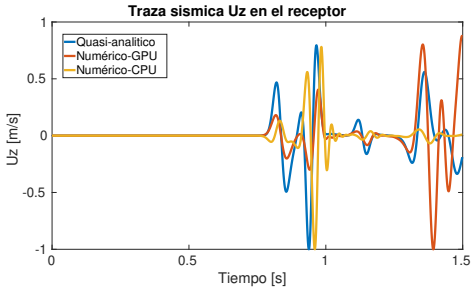
(a) 1 segundo de propagación



(b) 1 segundo de propagación



(c) 1,5 segundos de propagación



(d) 1,5 segundos de propagación

Figura 8: Trazas Sísmicas de velocidades para diferentes tiempos de propagación: Cada subfigura muestra el valor de los campos de velocidad en X (Ux)y velocidad en Z (Uz) en la posición del receptor para 1 segundo de propagación 8a, 8b, y 1,5 segundos de propagación 8c, 8d. Las graficas de cada subfigura representan los resultados en GPU, CPU y la solución quasi-analítica

Para cada una de las gráficas se determinó el error cuadrático medio y la correlación cruzada. Dando como resultados:

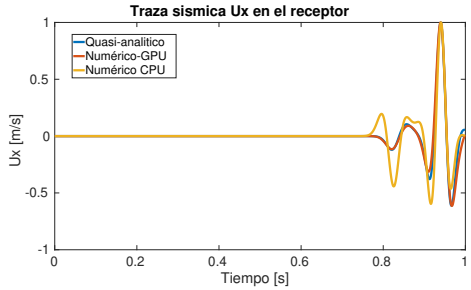
Traza Sísmica en GPU	**% Error Cuadrático Medio**	**% Correlación Cruzada**
Velocidad en X con 1 segundo de propagación (Figura 8a)	0,77 %	87,85 %
Velocidad en Z con 1 segundo de propagación (Figura 8b)	0,65 %	91,06 %
Velocidad en X con 1,5 segundos de propagación (Figura 8c)	0,85 %	82,81 %
Velocidad en Z con 1,5 segundos de propagación (Figura 8b)	4,70 %	34,05 %

Tabla 1: Error de trazas sísmicas respecto a respuesta quasi-analítica del propagador en GPU

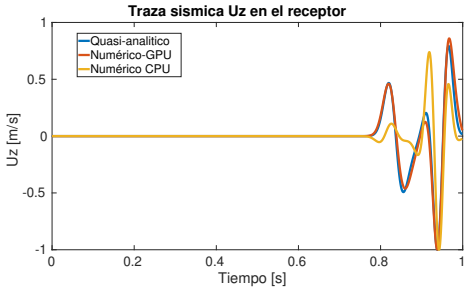
Traza Sísmica en CPU	**% Error Cuadrático Medio**	**% Correlación Cruzada**
Velocidad en X con 1 segundo de propagación (Figura 8a)	2,22 %	29,97 %
Velocidad en Z con 1 segundo de propagación (Figura 8b)	1,13 %	39,97 %
Velocidad en X con 1,5 segundos de propagación (Figura 8c)	1,52 %	6,618 %
Velocidad en Z con 1,5 segundos de propagación (Figura 8b)	5,05 %	31,41 %

Tabla 2: Error de trazas sísmicas respecto a respuesta quasi-analítica del propagador en CPU

Luego de obtener estos resultados, se disminuyó el paso de discretización de 10 metros a 5 metros con el objetivo de observar la disminución de error que esto causa y obtener un resultado más similar a la solución quasi-analítica. Los resultados de esta prueba se reflejan en la figura 9 y la tabla 3.



(a) 1 segundo de propagación



(b) 1 segundo de propagación

Figura 9: Trazas Sísmicas de velocidades para 1 segundo de propagación con $\Delta h = 5[m]$

Traza Sísmica en GPU	**% Error Cuadrático Medio**	**% Correlación Cruzada**
Velocidad en X con 1 segundo de propagación (Figura 9a)	0,06 %	98,68 %
Velocidad en Z con 1 segundo de propagación (Figura 9b)	0,11 %	98,67 %

Tabla 3: Error de trazas sísmicas respecto a respuesta quasi-analítica del propagador con $\Delta h = 5[m]$ durante una propagación de 1 segundo en GPU

Traza Sísmica en CPU	**% Error Cuadrático Medio**	**% Correlación Cruzada**
Velocidad en X con 1 segundo de propagación (Figura 9a)	2,18 %	31,65 %
Velocidad en Z con 1 segundo de propagación (Figura 9b)	6,13 %	8,51 %

Tabla 4: Error de trazas sísmicas respecto a respuesta quasi-analítica del propagador con $\Delta h = 5[m]$ durante una propagación de 1 segundo en CPU

La tendencia de los datos a no parecerse a la respuesta quasi-analítica, se puede explicar como una causa del operador de aproximación de diferencias finitas utilizado.

4.3. USO DE LA GPU

Para confirmar que el uso de GPU mejora la velocidad de procesamiento de los datos, se realizaron 3 pruebas que comparan la implementación en GPU contra una implementación en CPU, que contiene la misma discretización y la misma implementación de las fronteras absorbentes. Las pruebas constan de propagaciones sobre un modelo constante de 3000[m] de largo y 3000[m] de profundidad $\Delta h = 5[m]$, con $v_p = 2500[m/s]$, $v_s = 1558[m/s]$ y $\rho = 1500[Kg/m^3]$. En las cuales se cambio el tamaño del modelo o el tiempo de propagación.

Prueba	Tiempo en GPU [s]	Tiempo en CPU [s]	% Disminución
Nx=3000[m] Nz=3000[m] Tp=1,5 [s]	13,06034	52,840362	75,28 %
Nx=3000[m] Nz=3000[m] Tp=3,0 [s]	26,86936	101,98684	74,63 %
Nx=4000[m] Nz=4000[m] Tp=1,5 [s]	21,953672	92,502592	76,27 %

Tabla 5: Tiempos de ejecuciones de GPU y CPU para diferentes pruebas del algoritmo, cambiando el tamaño del modelo o el tiempo de propagación

Luego se probó la ventaja del cluster para procesar datos en paralelo haciendo uso de más de una GPU al tiempo. La prueba consiste en sobre el mismo modelo ubicar dos fuentes. Las fuentes en campo se explotan una después de otra, con un tiempo de separación entre cada disparo, suficiente como para que las ondas provocadas por el primer disparo sean atenuadas por el medio. Esto implica que no hay relación entre un disparo y otro y se pueden realizar en paralelo. Para esta prueba se utilizó un nodo del custer. Una de las GPUs del nodo ubicó la fuente en la posición x= 1000 [m] y z= 20[m], la segunda GPU ubica la fuente en x= 2000 [m] y z= 20[m]. Y luego ambas propagan.

Prueba	Tiempo en GPU [s]	Tiempo en CPU [s]	% Disminución
Nx=3000[m] Nz=3000[m] Tp=1,5 [s]	14,0126	102,0523	86,27 %

Tabla 6: Tiempos de ejecuciones de GPU y CPU con dos fuentes

En el desarrollo del proyecto se utilizó la herramienta *NVIDIA Visual Profiler*, la cual brinda información detallada de lo que ocurre dentro de la GPU cuando se ejecuta el programa. El objetivo de esta información es brindar métricas para posteriores optimizaciones. Algunos de los resultados obtenidos para la prueba sobre una GPU con 1 segundo de propagación son:

- Memoria de GPU utilizada: 45MiB

Luego se amplió el modelo a Nx=4000[m] y Nz=4000[m], con un consumo de RAM:

- Memoria de GPU utilizada: 60MiB

NVIDIA Visual Profiler organiza los kernels indicando de mayor a menor cual es el que mayor cantidad de recursos utilizó y por lo tanto el que con optimizaciones permitiría mejores rendimientos.

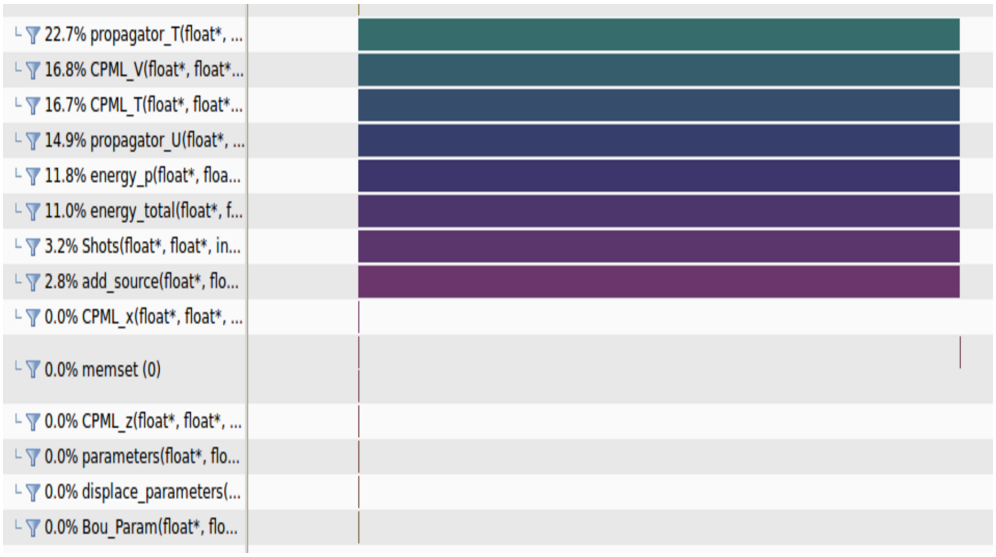


Figura 10: Resultados de herramienta Profiler: Cada una de las filas de la imagen representa el uso de un kernel. Se orden desde el kernel que mayor consume recursos, hasta el que menor consume.

Como se puede ver en la figura 10, el kernel que más recursos utiliza es el kernel $propagator\_T$ el cual implementa, de las ecuaciones de propagación 2.5, los campos τ_{xx}, τ_{zz} y τ_{xz} incluyendo las condiciones de CPML. Este kernel tiene 256 bloques con 112 *threads* cada uno, para un total de 28672 *threads*. Una métrica importante en los kernel es el número de registros por bloque, ya que si el cada *thread* utiliza muchos registros, se agotan los registros del *Streaming Multiprocessors* (SM) provocando que se necesite otro *Streaming Multiprocessors* para el otro bloque. Para dar un ejemplo de lo que ocurre en esta situación, cuando este kernel fue implementado con 72 registros por *thread*, se consumían 18432 registros por bloque 11.

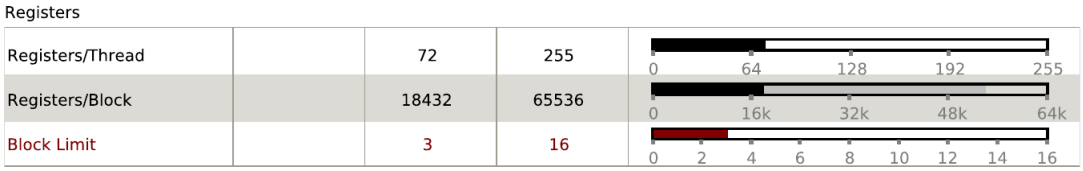


Figura 11: Resultados de herramienta Profiler: Kernerl de mayor consumo con 72 registros por *thread*: La tercera columna son los valores implementados, la cuarta columna son los valores disponibles y la ultima columna es una representación gráfica de ambos.

El número máximo de registros por SM es 65536 así que con 18432 registros por bloque, solo era posible implementar 3 bloques por SM, y como las GPUs Tesla K40 solo tienen 16 SMs eran necesarias varias ejecuciones para implementar los 256 bloques del kernel.

Una solución que brinda NVIDIA es que utilizando una bandera en el momento de compilar (–maxrregcount) podemos especificar la cantidad de registros por *thread*. A pesar que con esto podemos determinar el número de registros precisos para que se utilicen a plenitud cada *Streaming Multiprocessors* hay que tener cuidado con este valor, ya que si se llenan los SMs, el número de transacciones a memoria que tienen que hacer los SMs es mucho, aumentando el tiempo de ejecución. Por ejemplo si se utilizan 17 registros por *thread* (Figura 12), se realizan 514266 transacciones (Figura 13), mientras que para el para 30 registros por *thread* (Figura14) se realizan 200797 transacciones (Figura 15)

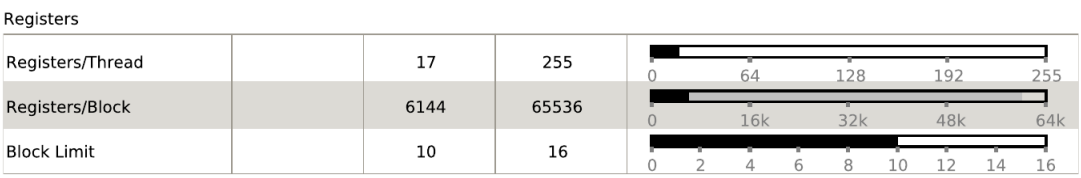


Figura 12: Resultados de herramienta Profiler: Kernerl de mayor consumo con 17 registros por *thread*: La tercera columna son los valores implementados, la cuarta columna son los valores disponibles y la ultima columna es una representación gráfica de ambos.

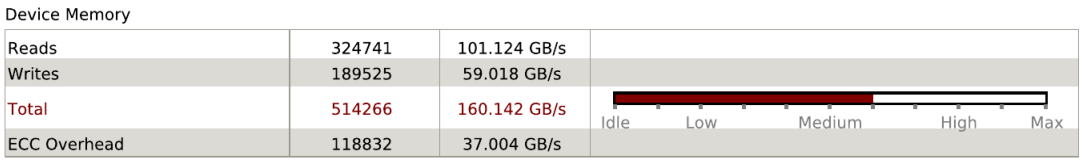


Figura 13: Resultados de herramienta Profiler: Kernerl de mayor consumo con 17 registros por *thread* (Transacciones a memoria): La segunda columna son las transacciones a la memoria de la GPU. la tercera columna es el valor del ancho de banda usado en las transacciones

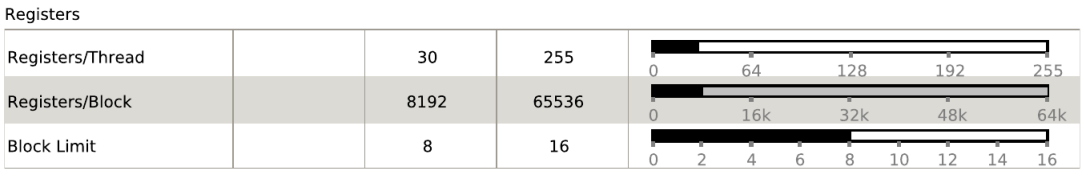


Figura 14: Resultados de herramienta Profiler: Kernerl de mayor consumo con 30 registros por *thread*: La tercera columna son los valores implementados, la cuarta columna son los valores disponibles y la última columna es una representación gráfica de ambos.

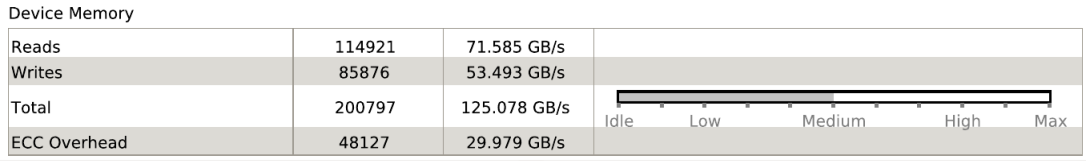


Figura 15: Resultados de herramienta Profiler: Kernerl de mayor consumo con 30 registros por *thread* (Transacciones a memoria): La segunda columna son las transacciones a la memoria de la GPU. la tercera columna es el valor del ancho de banda usado en las transacciones

Por esta razón no hay un número perfecto de registros que permita obtener un 100 % de rendimiento de la GPU, en todos los aspectos, pero dependiendo la aplicación es posible mejorar el aspecto más relevante.

4.4. DATOS DE LA PROPAGACIÓN

Uno de los objetivos planteados en el proyecto fue la construcción del código pensando en su futura utilización dentro de el esquema del método de inversión de onda completa(FWI). El diagrama mostrado en la figura 16 muestra una iteración de la FWI. Los cuadrados que estan en verde representan el algoritmo implementado,

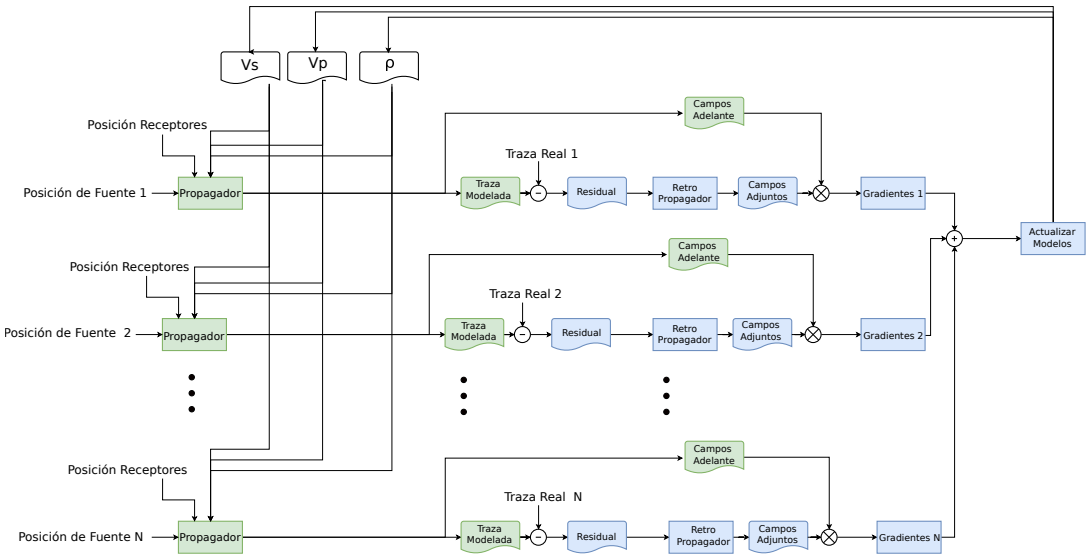


Figura 16: Diagrama del método de inversión de onda completa: en este diagrama se observa una iteración del método. Los cuadros verdes representas el algoritmo y las salidas desarrolladas en este trabajo. Las azules son los algoritmos restantes para ser implementados con el fin de lograr la inversión de onda completa elástica 2D.

y las salidas de este. Los cuadrados azules representan los algoritmos restantes para implementar la inversión de onda completa 2D elástica.

Los datos generados por el código (Trazas Modeladas y Trazas hacia adelante), se pueden observar en las figuras de la 17 a la figura 23. Todos los datos pertecen a la prueba realizada con el modelo de dos capas(Figura 7) con $\Delta h = 5[m]$ pero en lugar de un receptor, hay 600 receptores ubicados a 1090[m] de profundidad.

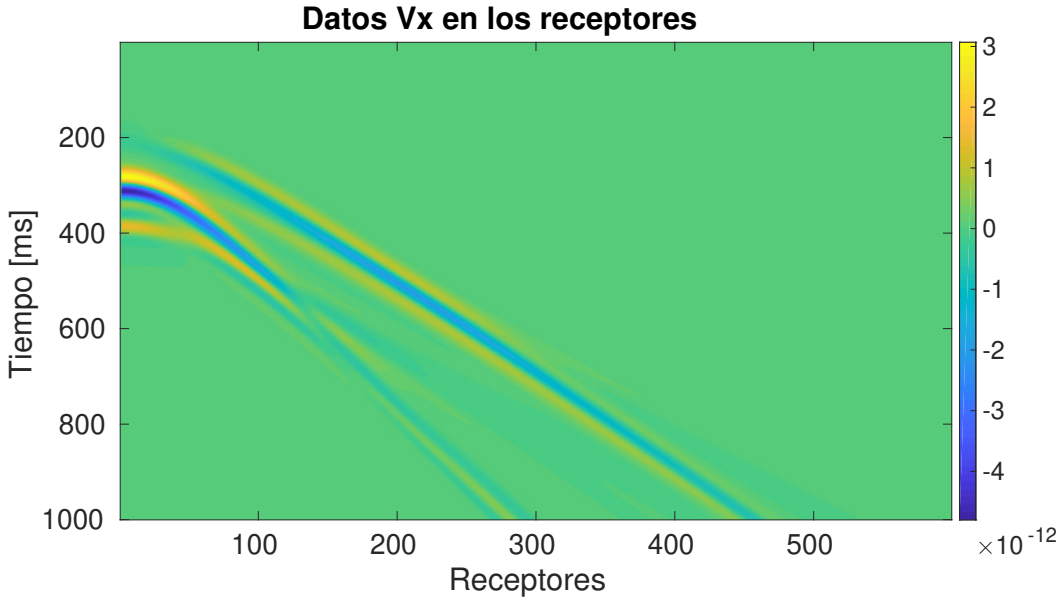


Figura 17: Datos en los 600 receptores para velocidad en X

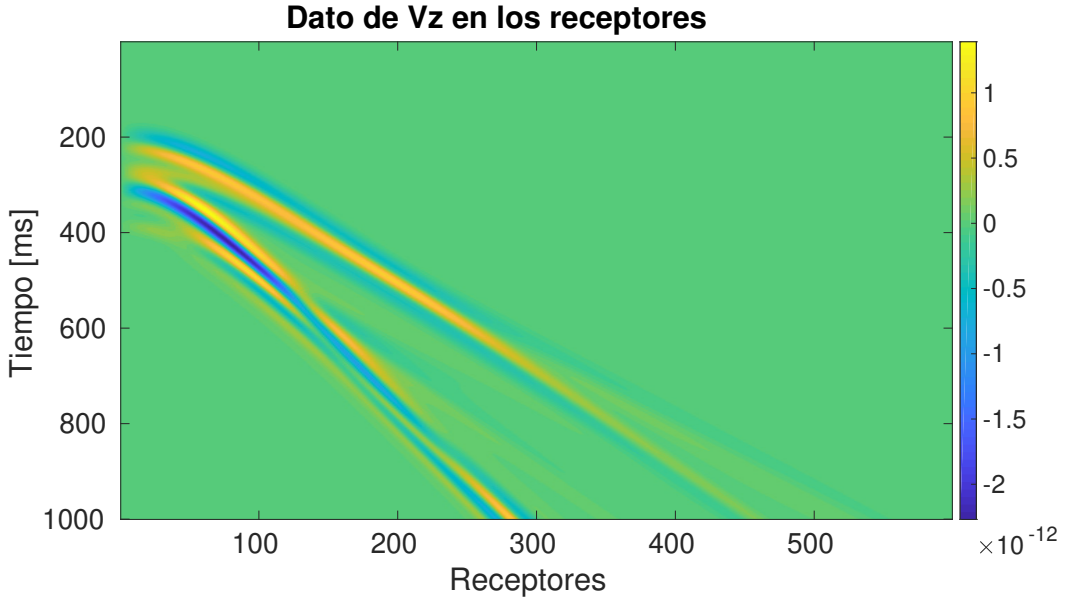


Figura 18: Datos en los 600 receptores para velocidad en Z

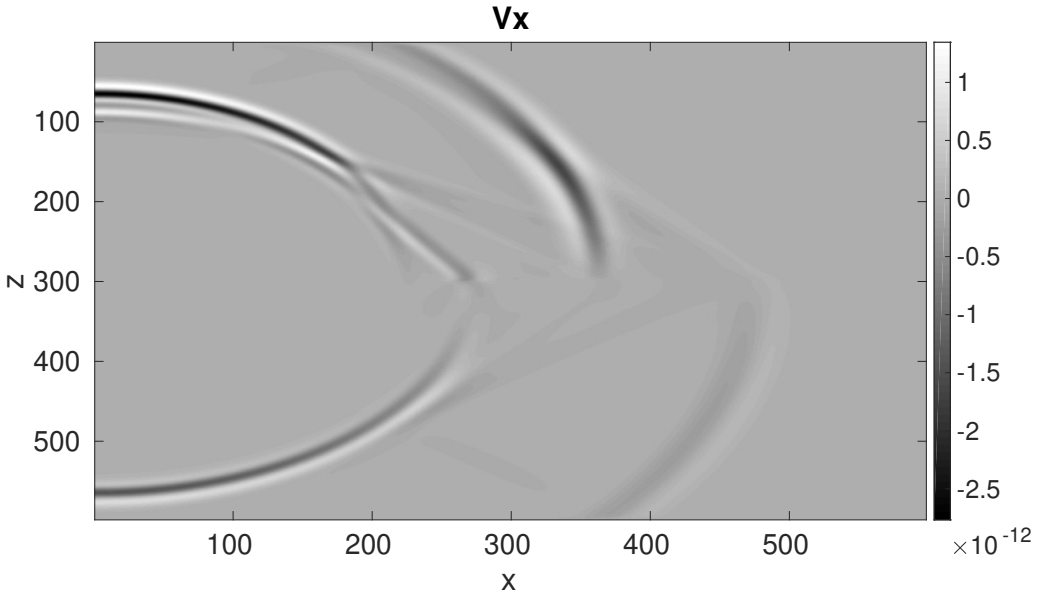


Figura 19: Campo Vx luego de 800 $[ms]$

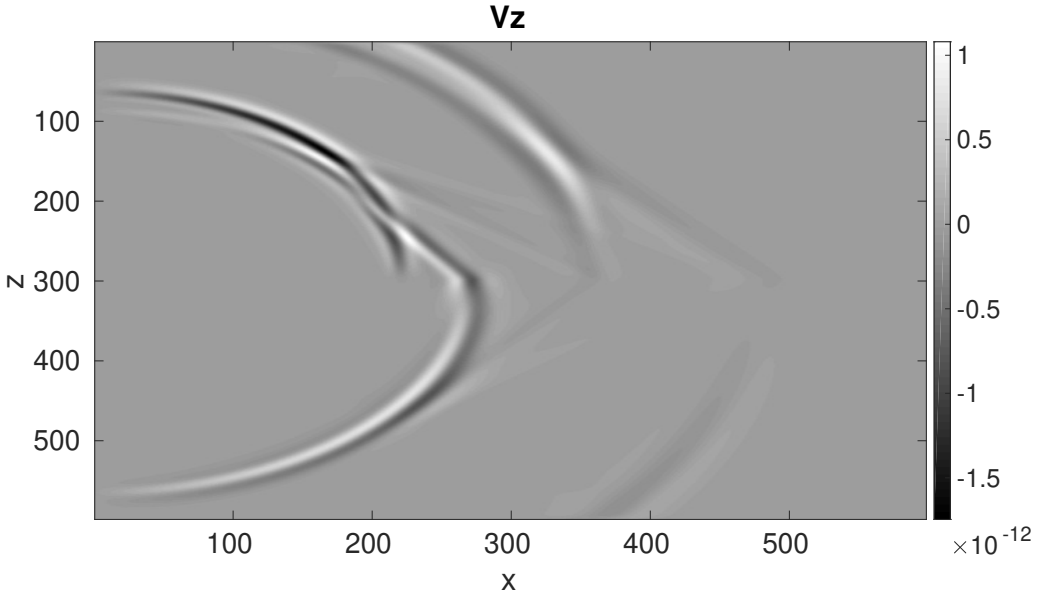


Figura 20: Campo Vz luego de 800 $[ms]$

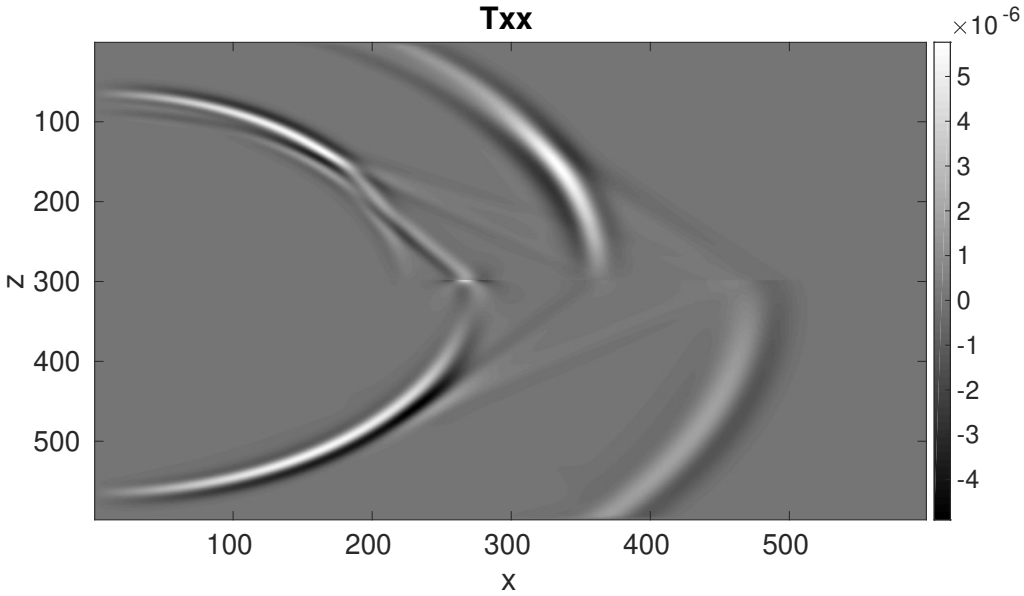


Figura 21: Campo Txx luego de 800 $[ms]$

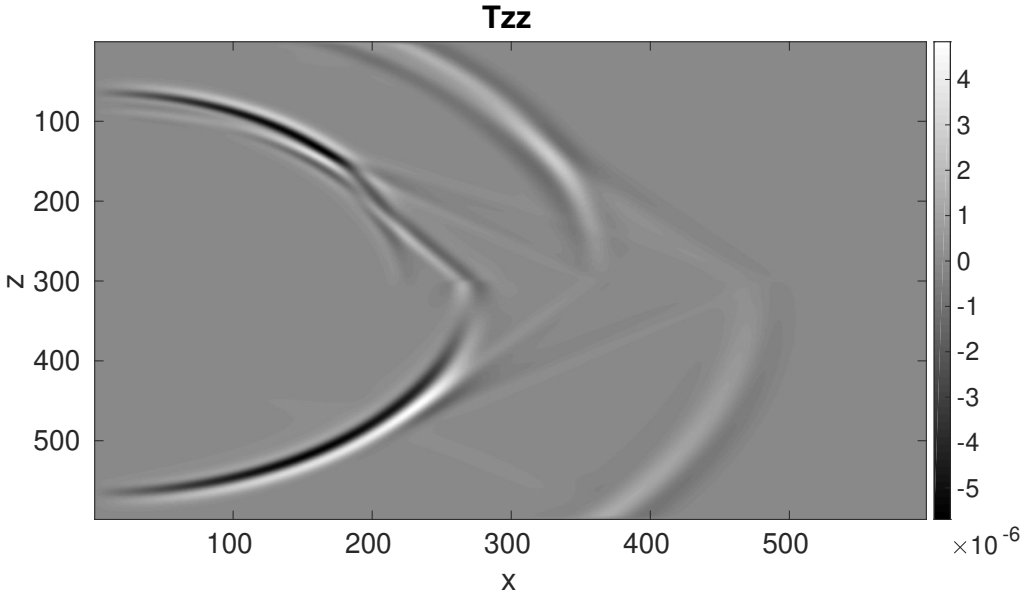


Figura 22: Campo Tzz luego de 800 $[ms]$

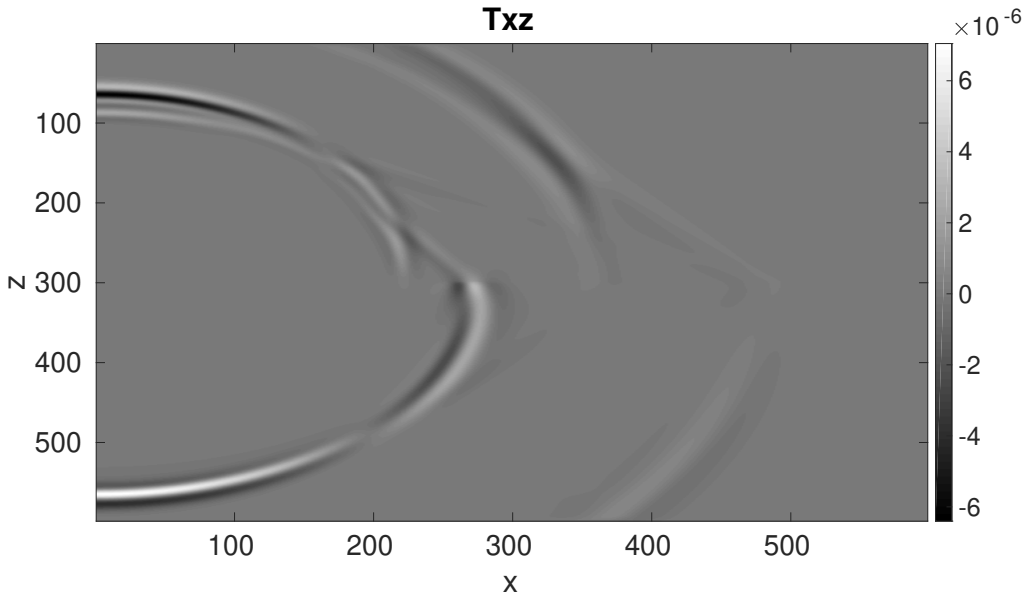


Figura 23: Campo Txz luego de 800 $[ms]$

5. CONCLUSIONES Y TRABAJO FUTURO

5.1. CONCLUSIONES

Se logró implementar el conjunto de ecuaciones que modelan la propagación de una onda elástica 2D en medio isótropo en una GPU. Se comprobó la implementación utilizando un algoritmo que permite calcular la quasi-analítica del problema de propagación elástica, esta solución se comparó con la implementada numericamente obteniendo un conjunto de errores para diferentes tiempos de propagación consignados en la tabla 1. Estos errores dependen de la aproximación en diferencias finitas utilizada y del paso de esta aproximación, como quedó demostrado cuando se disminuyó este paso (Figura 9). Es importante resaltar que estos errores serán o no significativos dependiendo de la aplicación que se le quiera dar al propagador.

Por otro lado se comprobó la absorción de la energía en las fronteras del modelo calculando la energía de la propagación y verificando que ésta disminuyera en función del tiempo cuando hay presencia de las ecuaciones de C-PML.

Respecto a la implementación, se ratificó que el rendimiento del algoritmo en GPU ofrece una disminución del tiempo respecto a la implementación en CPU, para nuestro caso una disminución de hasta 59 %. Y que al utilizar más de una GPU se obtienen mejoras de hasta 70 %, cuando se trabajan múltiples disparos. Claramente el algoritmo sigue siendo optimizable, haciendo un buen uso de variables, como el número de registros por kernel es posible llegar a tiempos menores.

5.2. TRABAJO FUTURO

Haciendo uso de las métricas mostradas en este trabajo y de un uso más detallado de herramientas como *NVIDIA Visual Profiler* es posible crear estrategias de optimización del código para mejorar su rendimiento. Cabe aclarar que para implementaciones de la propagación de onda las mejoras pueden no ser muy significativas si se trabajan con modelos y tiempos similares a los utilizados en este proyecto. Para modelos más grandes o implementaciones del algoritmo de inversión, estas mejoras resultan muy útiles. Otra estrategia a utilizar es el uso de la interfaz de paso de mensajes (MPI por sus siglas en ingles *Message Passing Interface*), el cual es

un estándar de comunicación de datos que permite utilizar recursos de diferentes GPUs como si se tratara de una misma. Lo que brindaría mayor número de recursos y la posibilidad de trabajar con datos mucho más grandes.

Utilizando el desarrollo matemático mostrado en [12], es posible crear los algoritmos que faltan del diagrama de la FWI(Figura 16, cuadrados azules) y obtener una implementación del algoritmo en GPU con el objetivo de reconstruir imágenes con información de la onda elástica. Adicionalmente es posible construir módulos de propagación elástica con anisotropía o utilizando un orden de aproximación de diferencias finitas mayor.

REFERENCIAS

[1] SLAWINSKI M. A., *Seismic Waves and Rays in Elastic Media*, vol. 34. Pergamon, 2003.

[2] KANE S. Yee, "Numerical solution of initial boundary value problems involving maxwell's equations in isotropic media," *IEEE TRANSACTIONS ON ANTENNAS AND PROPAGATION*, vol. 14, no. 3, pp. 302–307, 1966.

[3] VIRIEUX Jean., "Sh-wave propagation in heterogeneous media: Velocity-stress finite difference method," *Geophysics*, vol. 49, no. 11, pp. 1933–1957, 1984.

[4] RODEN, J.A. y GEDNEY, S.D., "Convolution pml (cpml): An efficient fdtd implementation of the cfs-pml for arbitrary media," *Microwave and Optical Technology Letters*, vol. 27, no. 5, pp. 334–339, 2000.

[5] VIRIEUX Jean., "P-sv wave propagation in heterogeneous media:velocity-stress finite-difference method," *Geophysics*, vol. 51, no. 4, pp. 889–901, 1986.

[6] DIMITRI Komatitsch and ROLAND Martin, "An unsplit convolutional perfectly matched layer improved at grazing incidence for the seismic wave equation," *Geophysics*, vol. 72, no. 5, pp. 155–167, 2007.

[7] Francis COLLINO and Chrysoula TSOGKA, "Application of the perfectly matched absorbing layer model to the linear elastodynamic problem in anisotropic heterogeneous media," *Geophysics*, vol. 66, no. 1, pp. 294–307, 2001.

[8] COURANT, Richard., FRIEDRICHS Kurt. and LEWY Hans., "On the partial difference equations of mathematical physics," *IBM Journal*, vol. 11, no. 2, pp. 215–234, 1967.

[9] "Métodos geofísicos." `http://www.medellin.unal.edu.co/~rrodriguez/geologia/sismica.htm`. [Online].

[10] "CUDA C Programming Guide." `http://docs.nvidia.com/cuda/`, 2015. [Online].

[11] "EX2DELEL." `http://www.spice-rtn.org/library/software/EX2DELEL.html`. [Online].

[12] WESDORP Jaap, "A general recipe for obtaining adjoint equations and gradients and its application to full waveform inversion of 2d isotropic elastic media in finite difference time-domain." Junio 2018.

BIBLIOGRAFÍA

COURANT, Richard., FRIEDRICHS Kurt. and LEWY Hans., "On the partial difference equations of mathematical physics," *IBM Journal*, vol. 11, no. 2, pp. 215–234, 1967

"CUDA C Programming Guide." http://docs.nvidia.com/cuda/, 2015. [Online]

DIMITRI Komatitsch and ROLAND Martin, "An unsplit convolutional perfectly matched layer improved at grazing incidence for the seismic wave equation," *Geophysics*, vol. 72, no. 5, pp. 155–167, 2007

"EX2DELEL." http://www.spice-rtn.org/library/software/EX2DELEL.html. [Online]

Francis COLLINO and Chrysoula TSOGKA, "Application of the perfectly matched absorbing layer model to the linear elastodynamic problem in anisotropic heterogeneous media," *Geophysics*, vol. 66, no. 1, pp. 294–307, 2001

KANE S. Yee, "Numerical solution of initial boundary value problems involving maxwell's equations in isotropic media," *IEEE TRANSACTIONS ON ANTENNAS AND PROPAGATION*, vol. 14, no. 3, pp. 302–307, 1966

"Métodos geofísicos." http://www.medellin.unal.edu.co/~rrodriguez/geologia/sismica.htm. [Online]

RODEN, J.A. y GEDNEY, S.D., "Convolution pml (cpml): An efficient fdtd implementation of the cfs-pml for arbitrary media," *Microwave and Optical Technology Letters*, vol. 27, no. 5, pp. 334–339, 2000

SLAWINSKI M. A., *Seismic Waves and Rays in Elastic Media*, vol. 34. Pergamon, 2003

VIRIEUX Jean., "P-sv wave propagation in heterogeneous media:velocity-stress finite-difference method," *Geophysics*, vol. 51, no. 4, pp. 889–901, 1986

VIRIEUX Jean., "Sh-wave propagation in heterogeneous media: Velocity-stress finite

difference method," *Geophysics*, vol. 49, no. 11, pp. 1933–1957, 1984

WESDORP Jaap, "A general recipe for obtaining adjoint equations and gradients and its application to full waveform inversion of 2d isotropic elastic media in finite difference time-domain." Junio 2018