JOURNAL
CIENCIA, TECNOLOGÍA Y FUTURO

ISSN (Print) 0122-5383
ISSN (Online) 2382-4581
Journal of oil, gas and alternative energy sources

A PRACTICAL IMPLEMENTATION OF ACOUSTIC FULL WAVEFORM INVERSION ON GRAPHICAL PROCESSING UNITS

UNA IMPLEMENTACIÓN PRÁCTICA DE LA INVERSIÓN DE ONDA COMPLETA ACÚSTICA SOBRE UNIDADES DE PROCESAMIENTO GRÁFICO

UMA IMPLEMENTAÇÃO PRÁTICA DA INVERSÃO DE ONDA COMPLETA ACÚSTICA SOBRE UNIDADES DE PROCESSAMENTO GRÁFICO

Sergio-Alberto Abreo-Carrillo[1*], Ana-B. Ramirez[1], Oscar Reyes[1],
David-Leonardo Abreo-Carrillo[1] and Herling González-Alvarez[2]

[1]CPS Research Group, Depart. Electrical, Electronic and Telecom. Eng., Universidad Industrial de Santander, Bucaramanga, Santander, Colombia
[2]Ecopetrol S.A. - Instituto Colombiano del Petróleo (ICP), A.A. 4185 Bucaramanga, Santander, Colombia

e-mail: abreosergio@gmail.com

(Received: Apr. 24, 2015; Accepted: Nov. 26, 2015)

ABSTRACT

Recently, Full Waveform Inversion (FWI) has gained more attention in the exploration geophysics community as a data fitting method that provides high-resolution seismic velocity models. Some of FWI essential components are a cost function to measure the misfit between observed and modeled data, a wave propagator to compute the modeled data and an initial velocity model that is iteratively updated until an acceptable decrease of the cost function is reached.

Since FWI is a wave equation based method, the computational costs are elevated. In this paper, it is presented a fast Graphical Processing Unit (GPU) FWI implementation that uses a 2D acoustic wave propagator in time and updates the model using the gradient of the cost function, which is efficiently computed with the adjoint state method. The proposed parallel implementation is tested using the Marmousi velocity model. The performance of the proposed implementation is evaluated using the NVIDIA GeForce GTX 860 GPU and compared to a serial Central Processing Unit (CPU) implementation, in terms of execution time. We also evaluate the GPU occupancy and analyze the memory requirements. Our tests show that the GPU implementation can achieve a speed-up of 26.89 times when compared to its serial CPU implementation.

Keywords: *Full waveform inversion, Adjoint state method, Graphical processing units, Convolutional perfect matched layer, Seismic modeling, Wave propagation.*

How to cite: Abreo-Carrillo, S. A., Ramirez, A. B., Reyes, O., Abreo-Carrillo, D. L. & González-Alvarez, H. (2015). A practical implementation of acoustic full waveform inversion on graphical processing units. *CT&F - Ciencia, Tecnología y Futuro*, 6(2), 5-16.

*To whom correspondence should be addressed

RESUMEN

Recientemente, la inversión de onda completa (FWI, por sus siglas en inglés) ha ganado una mayor atención en la comunidad de exploración geofísica como un método de ajuste de datos que provee modelos de velocidades sísmicas de gran resolución. Algunos de los componentes esenciales de la FWI corresponden a una función de costo para medir la diferencia entre los datos observados y los datos modelados, un propagador de onda para obtener los datos modelados y un modelo de velocidad inicial que es actualizado iterativamente hasta llegar a un valor deseado de la función de costo.

Como la FWI es un método basado en la ecuación de onda, el costo computacional de su implementación es elevado. En este documento presentamos una implementación rápida de la FWI 2D acústica en tiempo sobre una unidad de procesamiento gráfico (GPU, por sus siglas en inglés). Esta implementación usa la ecuación de onda acústica para modelar la propagación y actualiza el modelo de velocidades usando el gradiente de la función de costo, el cual es calculado eficientemente usando el Método del Estado Adjunto. La implementación paralela propuesta es probada usando el modelo de velocidades Marmousi. El desempeño de la implementación propuesta es evaluado usando una GPU NVIDIA GeForce GTX 860 y comparado con una implementación serial sobre un procesador, en términos de tiempo de ejecución. Adicionalmente, se evalúa la cantidad de recursos usados por la GPU y se analizan los requerimientos de memoria de la implementación. Las pruebas muestran que la implementación sobre GPU puede alcanzar un índice de aceleración de 26.89 veces si se compara con la implementación serial sobre el procesador.

Palabras clave: *Inversión de onda completa, Método del estado adjunto, Unidades de procesamiento gráfico, Capas perfectamente emparejadas, Modelamiento sísmico, Propagación de onda.*

RESUMO

Recentemente, a inversão de onda completa (FWI, sigla em inglês) ganhou maior atenção na comunidade de exploração geofísica como método de ajuste de dados, que fornece modelos de velocidades sísmicas de alta resolução. Alguns dos componentes essenciais do FWI são uma função de custo para estimar a diferença entre os dados observados e os dados modelados, um propagador do campo de ondas acústicas para os dados modelados e um modelo de velocidade inicial, que é atualizada de forma iterativa.

Como o FWI está baseado no método da equação da onda, as exigências computacionais de execução são altas. Neste artigo apresentamos uma implementação rápida do FWI acústico 2D em tempo em uma unidade de processamento gráfico (GPU, sigla em inglês). Esta implementação utiliza um propagador da equação de onda e atualiza o modelo de velocidade, utilizando o gradiente da função objetivo, que é calculada de forma eficiente usando o método do estado adjunto. Proposta de implementação paralela é testada utilizando o modelo de velocidade Marmousi. O desempenho da implementação proposta é avaliada usando uma GeForce GTX 860 e comparada com uma aplicação de série em, um único processador, em termos de tempo de execução. Avaliamos também a quantidade de recursos utilizados pela GPU e analisamos os requisitos de memória. Os testes mostram que a implementação em GPU pode conseguir uma taxa de aceleração de 26.89 vezes quando comparada com uma implementação serial do processador.

Palavras-chave: *Processo de inversão de forma de onda completa, Método adjunto, Unidades de processamento gráfico, Modelagem sísmica, Propagação de ondas.*

1. INTRODUCTION

Full Waveform Inversion (FWI) has gained considerable attention in the geophysics community in the last years due to its capabilities to generate seismic velocity models with high resolution. Although its theory was developed by Lailly (1983) and Tarantola (1984) around 1983, FWI has not been widely applied to real problems because of some difficulties, such as: non-uniqueness of the solution, its dependency on a good initial model and its high computational cost. This problem is becoming more tractable due to the evolution of computing systems, and FWI is now being applied to obtain subsurface parameters for both short- and long-offset acquisitions (Virieux & Operto, 2009; Vigh *et al*., 2014). Recent works have shown the capabilities of FWI using traditional computer architectures (Bunks *et al*., 1995; Thierry, Donno & Noble, 2014; Etienne *et al*., 2014; Cao & Liao, 2014). Furthermore, as a consequence of the development of parallel computer architectures such as Graphics Processing Units (GPUs), some studies have targeted to implementing FWI in time domain using GPUs, achieving speedups in the computation time in comparison to its serial implementation counterpart (Wang *et al*., 2011; Mao, Wu & Wang, 2012; Kim, Shin & Calandra, 2012; Weiss & Shragge, 2013; Zhang *et al*., 2014; Yang, Gao & Wang, 2015).

For example, Weiss and Shragge (2013) proposed a parallel implementation for solving the 2D and 3D elastic wave equation on GPU devices, achieving speedups of 10 and 28 times, respectively. The main differences between Weiss's work and our work are: 1) Weiss's work only evaluates the Finite Difference in Time Domain (FDTD) implementation inside the GPU whereas our work not only computes the FDTD but also the gradient, and the update of the velocity model inside the GPU. 2) In Weiss's work, the elastic FDTD is evaluated, whereas in our work we solve the acoustic wave equation. 3) Also, the order of the space discretization in the FDTD is different (Weiss used 8th order in space, we used 2nd order). 4) Finally, the GPU used in their work is the NVIDIA GTX 480x, whereas we used the NVIDIA GeForce GTX 860M.

This work proposes a practical parallel implementation of FWI in time on a GPU architecture, using an algorithm where the modeled wavefield is computed inside the GPU, avoiding CPU-GPU data transfer. Although this implementation allows us to reduce the computation time of FWI, the amount of GPU RAM required to store the entire wavefield becomes a constraint as the size of the velocity model and the number of shots increase. In order to compute the modeled wavefield, we use a finite-differences discretization of second order in time and space of the 2D acoustic wave equation. Furthermore, the Convolutional Perfect Matched Layer (CPML) method is used to model the absorbing boundary conditions. The gradient of the cost function, obtained with the adjoint state method, and the update of the velocity model are also computed inside the GPU. The Marmousi velocity model (Versteeg & Grau, 1991) is used to test the performance of the proposed GPU implementation.

The main contributions of this work are twofold. First, we propose a GPU implementation algorithm of FWI in the time domain from a practical point of view, such that it can be easily reproduced. Second, we present a performance analysis of the proposed implementation, not only in terms of execution time, but also in terms of GPU occupancy and RAM. The last two performance parameters are required to evaluate the capabilities of the implementation when the problem is scaled up. The outline of this paper is as follows. In section two, we present the theoretical framework of the wave propagation model. In section three, we describe the parallel implementation of FWI in GPU. Experimental results are presented in section four. Finally, discussion and conclusions are presented in sections five and six, respectively.

2. WAVE PROPAGATION MODELING

This subsection describes the 2D isotropic acoustic wave equation used to model the propagation of a source through the medium and, the implementation of the CPML technique to absorb the energy on boundaries.

Isotropic Acoustic Wave Equation

In particular, we model the two-dimensional wave propagation using the acoustic and isotropic wave equation defined by

$$\frac{1}{v^2(x,z)}\frac{\partial^2 p}{\partial t^2} = \frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial z^2} + src(x,z), \qquad (1)$$

where $v(x, z)$ is the acoustic velocity of the medium as a function of the spatial variables x and z, p denotes the scalar pressure field, and t is the time variable. A discretized version of *Equation 1* corresponding to a second-order finite-difference approximation in space and time (Alford, Kelly & Boore, 1974) is given by

$$p^{n+1}_{i,j} = 2(1-2C^2)p^n_{i,j} - p^{n-1}_{i,j} + C^2\left(p^n_{i+1,j} + p^n_{i-1,j} + p^n_{i,j+1} + p^n_{i,j-1}\right) \qquad (2)$$

where $C = \dfrac{v(x,z)\cdot\Delta t}{\Delta h} \leq \dfrac{1}{\sqrt{2}}$, i and j denote the discrete spatial variables, Δh is the spatial step, Δt is the time step, and n denotes the discrete time variable.

A graphical representation of the stencil used to implement the *Equation 2* is shown in Figure 1. A single point, $p^{n+1}_{i,j}$, in the propagated wave field in the future (Figure 1c), requires information of five points of the present (Figure 1b), and one point of the past (Figure 1a) wave fields, which are supposed to be known.

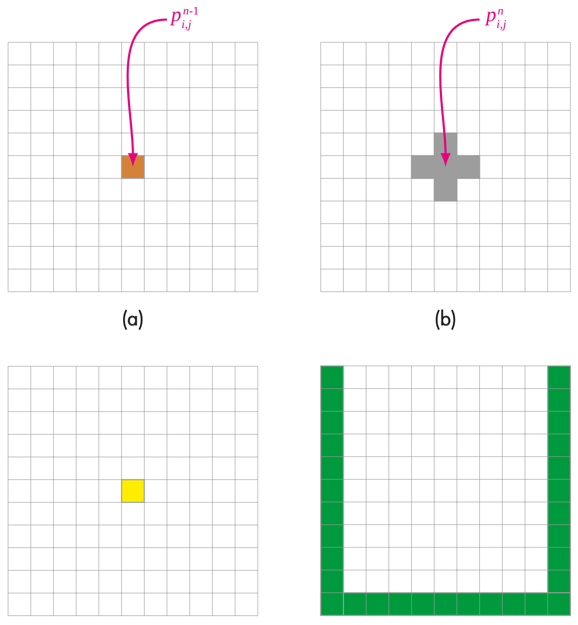


Figure 1. Stencil for computing the scalar field p at a spatial point located outside the boundaries area. (a) Past wave field (b) Present wave field (c) Future wave field (d) Boundaries area.

Convolutional Perfectly Matched Layer

The boundary conditions are required to minimize artificial reflections inside the area of interest. Several methods have been proposed to include the boundary conditions in the solution of the wave equation, such as Perfectly Matched Layer (PML), proposed by Berenger (1994); Nearly Perfectly Matched Layer (NPML), proposed by Hu, Abubakar and Habashy (2007); and Convolutional Perfectly Matched Layer (CPML), proposed by Pasalic and McGarry (2010). We use CPML since it has proven to be an efficient method when compared to PML, and offers good energy absorption at the boundaries. The isotropic acoustic wave equation including CPML requires two auxiliary variables at each spatial dimension. The modified version of the wave equation is given by

$$\frac{1}{v(x,z)^2}\frac{\partial^2 p}{\partial t^2} = \frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial z^2} + \frac{\partial \psi_x}{\partial x} + \frac{\partial \psi_z}{\partial z} + \zeta_x + \zeta_z. \qquad (3)$$

A discretized version of the continuous acoustic wave shown in *Equation 3*, including CPML, is given by

$$\begin{aligned} p^{n+1}_{i,j} = {} & 2(1-2C^2)\,p^n_{i,j} - p^{n-1}_{i,j} + C^2\left(p^n_{i+1,j} + p^n_{i-1,j} + p^n_{i,j+1} + p^n_{i,j-1}\right) + \\ & C^2\Delta h \cdot \psi^n_{st} + (C\cdot\Delta h)^2\zeta^n_{st}, \end{aligned} \qquad (4)$$

where
$$\psi^n_{st} = \psi^n_{x,i+1,j} - \psi^n_{x,i,j} + \psi^n_{z,i,j+1} - \psi^n_{z,i,j},$$
$$\zeta^n_{st} = \zeta^n_{x,i,j} + \zeta^n_{z,i,j}.$$

According to Pasalic and McGarry (2010), the auxiliary variables that minimize the artificial reflections at the boundaries are given by

$$\psi^n_q = b_q\psi^{n-1}_q + a_q\frac{\partial p}{\partial q}, \qquad (5)$$

$$\zeta^n_q = b_q\zeta^{n-1}_q + a_q\left[\left(\frac{\partial^2 p}{\partial q^2}\right)^n + \left(\frac{\partial \psi_q}{\partial q}\right)^n\right], \qquad (6)$$

where q can be either x or z, ψ^0_q and ζ^0_q are zero at the first iteration, and a_q and b_q can be found using *Equations 7* to *10* (Collino & Tsogka, 2001).

$$a_x = \frac{d_x}{d_x - \alpha_x}(b_x - 1), \qquad (7)$$

$$b_x = e^{-(d_x+\alpha_x)\Delta t}, \qquad (8)$$

$$d_x = d_0 V_{max}\left(\frac{f(x)}{Lx}\right)^2, \qquad (9)$$

$$\alpha_x = \pi f\left(\frac{Lx - f(x)}{Lx}\right). \qquad (10)$$

In this work, the parameters used to compute the auxiliary variables are

$$f(x) = \begin{cases} Lx : \Delta h : 0; & x \in [0, CPML - 1] \\ 0; & x \in [CPML, Nx - CPML] \\ 0 : \Delta h : Lx; & x \in [Nx - CPML + 1, Nx], \end{cases}$$

where $Lx = CPML \cdot \Delta h$, $CPML = 20$, $\Delta h = 25$ m, $d_0 = \frac{-3}{2Lx}\log(R)$, $R=0.001$, and $V_{max} = 5000$ m/s.The graphical representation of a stencil used to compute a spatial point located on the boundaries area, is shown in Figure 2.

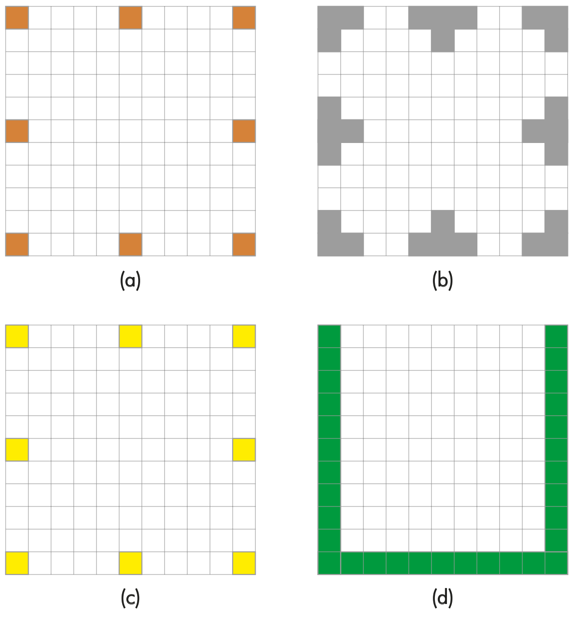


Figure 2. Stencil for computing scalar field *p* at specific spatial points located in the boundaries area. (a) Past wave field (b) Present wave field (c) Future wave field (d) Boundaries area.

3. FWI METHOD AND GPU IMPLEMENTATION

In the previous section, we described the acoustic and isotropic wave equation, which is used as the forward operator in FWI method. In this section, we describe the cost function and the updating strategy of the seismic velocity model. Also an algorithm is presented along with a description of the GPU implementation.

Full Waveform Inversion (FWI)

FWI is a non-linear inversion method that iteratively estimates subsurface properties, such as the seismic velocity. The main components of FWI are: a cost function to measure the misfit between the observed and modeled data, a wave propagator to compute the modeled data and an initial velocity model that is iteratively updated until the cost function reaches an adequate value.

Usually the misfit function is the least squares error function given by $\Phi(\mathbf{v}) = \frac{1}{2} \| G(\mathbf{v}) - \mathbf{d}^{obs} \|_2^2$, where G is the forward non-linear operator, $\mathbf{v} \in \mathbf{R}^{Nx \cdot Nz}$ is the velocity model, and $\mathbf{d}^{obs}$ is the data recorded at the surface. The problem of estimating $\mathbf{v}$ from $\mathbf{d}^{obs}$ is an ill-posed problem, and the solution is non-unique. From an optimization point of view, the misfit function has a global minimum and more than one local minimum. Thus, the convergence towards the global minimum, when using an iterative inverse method, depends on the selected starting velocity model.

Let $\mathbf{v} \in \mathbf{R_g}^{Nx \cdot Nz}$ be the vector containing the velocities of the model, for all x and z. The cost function $\Phi(\mathbf{v})$ is a scalar that measures the error between the observed and the modeled data. In this work, we use the ℓ_2^2 error norm as cost function, which is given by

$$\Phi(\mathbf{v}) = \frac{1}{2} \| \mathbf{d}(\mathbf{x} = x, z = 75 \text{ m}, \mathbf{t}|\mathbf{v}) - \mathbf{d}^{obs}(\mathbf{x} = x, z = 75 \text{ m}, \mathbf{t}) \|_2^2, \quad (11)$$

where $\mathbf{d}^{obs}(\mathbf{x} = x, z = 75 \text{ m}, \mathbf{t})$ represents the observed data in vectorized form at a given spatial location ($\mathbf{x} = x$, $z =$ 75 m), and $\mathbf{d}(\mathbf{x} = x, z = 75 \text{ m}, \mathbf{t}|\mathbf{v})$ represents the modeled data in vectorized form at the same spatial coordinates using a known velocity model $\mathbf{v}$. The modeled data in *Equation 11*, is obtained using the 2D acoustic and isotropic wave equation. Its numerical solution is given by $\mathbf{d}(\mathbf{x} = x, z = 75 \text{ m}, \mathbf{t}|\mathbf{v}) = \mathbf{p}(\mathbf{x} = x, z = 75 \text{ m}, \mathbf{t})$, where $\mathbf{p}(\mathbf{x} = x, z = 75 \text{ m}, \mathbf{t})$ is the scalar pressure field obtained with *Equation 4.*

An update for the velocity model can be obtained using steepest descent method as

$$\mathbf{v}^{k+1} = \mathbf{v}^k - \alpha \cdot \mathbf{g}(\mathbf{v}^k), \quad (12)$$

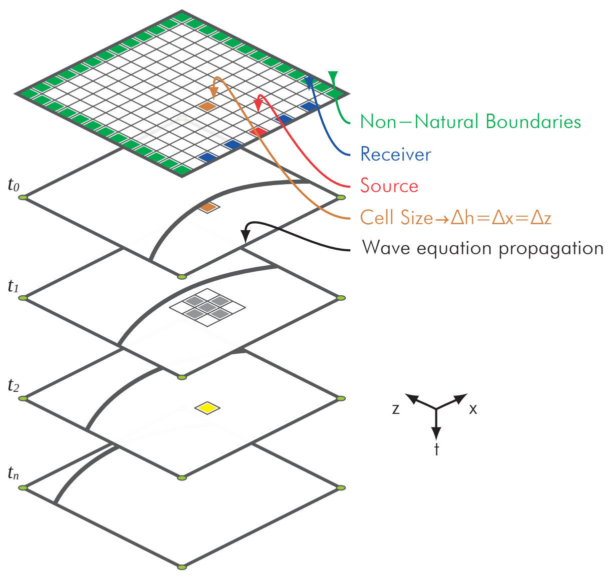
where $\mathbf{g}(\mathbf{v}^k)$ is the gradient of the cost function, obtained with the adjoint state method (Plessix, 2006). In order to find an adequate parameter α, we ran a set of experiments for different step size values and we select the one that retrieves the minimum error norm after a certain number of iterations. This search over different values of α is computationally expensive, and instead this parameter can be estimated using a line-search method, as proposed by Tarantola (1987). The gradient is given by

$$\mathbf{g}(\mathbf{v}^k) = -\sum_s \frac{2}{(v^k(\mathbf{x},\mathbf{z}))^3} \int_0^T q_s(\mathbf{x},\mathbf{z},T-t) \frac{\partial^2 p_s(\mathbf{x},\mathbf{z},t)}{\partial t^2} dt, \qquad (13)$$

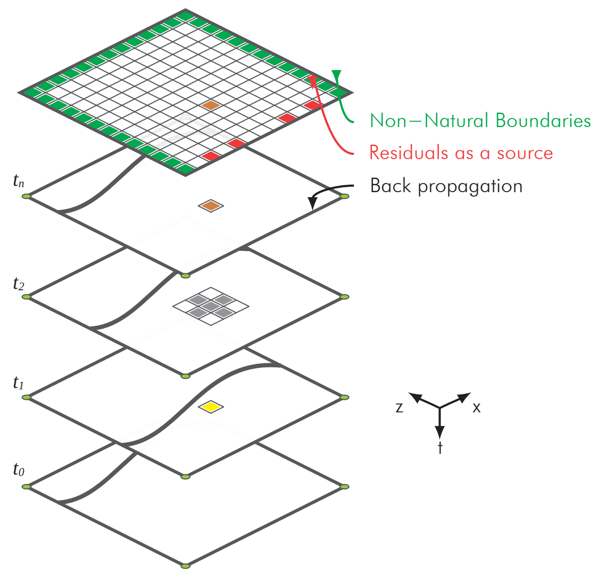
where the wave field q_s is obtained with *Equation 4* having as source the error between the modeled and observed data, in reverse time. For this reason, q_s is called the back-propagated field of the residual.

Figure 3a shows the propagation of the pressure field generated by a source. Note that the receivers record the pressure wave field at the surface level when a single source is used.

The residual between the modeled and observed seismic traces is then used as a source to find the back-propagated field, $q_s(\mathbf{x}, \mathbf{z}, \mathbf{t})$. This process is shown in Figure 3b. The main difference between Figures 3a and 3b is that in Figure 3b, the original source disappears and each receiver becomes a source whose signature is a residual. The curved lines in Figure 3 illustrate an example of the wavefront propagation of the sources. Having both wavefields, *Equation 13* is then used to compute the gradient $\mathbf{g}(\mathbf{v}^k)$.



(a)



(b)

Figure 3. (a) Propagation of a source (in red) at different time steps using the wave equation. The receivers are depicted in blue and the boundaries area is depicted in green. (b) Backpropagation of the residuals (in red) at different time steps using the wave equation. The boundaries area is depicted in green.

GPU Implementation

A kernel is a function used to run applications inside a GPU. When a kernel is launched, it uses blocks with a certain number of threads. A thread represents the minimum group of hardware elements required to run a task. All the threads within the same block work in parallel, and the number of blocks working in parallel depends on the number of streaming multiprocessors (SMs) inside the GPU. Usually, a kernel requires various blocks to execute a function and, depending on the number of SMs, the function is partially or completely parallelized.

We assign *Equation 4* to each thread to calculate every future point of the wavefield in parallel (see yellow points in Figures 1, 2 and 3). The total number of future points that can be computed in parallel in the NVIDIA GeForce GTX 860M are 10240. This occurs since the GPU has 1024 threads per block, two blocks per SMs and five SMs (1024 × 2 × 5, see Table 2). A pseudocode of the implementation is presented in Algorithm 1.

Algorithm 1 Full Waveform Inversion implemented inside a GPU

```
1:  $FWI\ (v, Obs, s, inf, \alpha)$                              ▹FWI inputs
2:  $v$                                                  ▹Starting velocity model
3:  $Obs$                                                ▹Observed traces
4:  $s$                                                  ▹Wavelet source
5:  $inf$                                   ▹Location and offset of each shot
6:  $\alpha$                                                  ▹Alpha value
7:  for $i \leftarrow 1, iG$ do                  ▹iG, Number of FWI interations
8:      $acum \leftarrow 0$
9:      $g(x, z) = 0$
10:     for $j \leftarrow 1, Ns$ do                     ▹Ns, Number of sources
11:         for $t \leftarrow 1, Nt$ do                 ▹Nt, Number of time steps
12:             $\frac{1}{v(x,z)^2}\frac{\partial^2 p_s}{\partial t^2} = \frac{\partial^2 p_s}{\partial x^2} + \frac{\partial^2 p_s}{\partial z^2} + S(sj,t)$
13:             $Mod\ (x,t) \leftarrow p_s\,(x,0,t)$
14:         end for
15:         $Obs_{dev} \leftarrow Obs_{j,host}$   ▹Observed traces from host to device
16:         $Residual(x,t) \leftarrow Mod\ (x,t) - Obs_{dev}(x,t)$
17:         $norm \leftarrow ||Residual(x,t)||_2^2$                  ▹(L2 Norm)$^2$
18:         $acum \leftarrow \frac{1}{2} norma + acum$
19:         for $t \leftarrow 1, Nt$ do
20:             $\frac{1}{v(x,z)^2}\frac{\partial^2 q_s}{\partial t^2} = \frac{\partial^2 q_s}{\partial x^2} + \frac{\partial^2 q_s}{\partial z^2} + Residual\ (x,Nt-t)$
21:         end for
22:         $g(x,z) = g(x,z) - \frac{2}{(v^k(\mathbf{x},\mathbf{z}))^3}\int_0^T q_s(x,z,T-t)\frac{\partial^2 p_s(x,z,t)}{\partial t^2}\,dt$   ▹
    Equation 13
23:     end for
24:     $\phi(i) \leftarrow acum$
25:     $v(x,z) \leftarrow v(x,z) - \alpha \cdot g(x,z)$      ▹Model update. Equation 12
26: end for
27: $V_{end} \leftarrow v(x,z)$            ▹Final velocity model from device to host
28: $\phi_h \leftarrow \phi$       ▹Objective function evolution from device to host
29: return $\phi_h, v_{end}$
```

4. EXPERIMENTAL RESULTS

In this section, we describe the velocity model and the geometry used to test the FWI implementation. Also, we present the execution time required by our implementation, and the hardware characteristics used in all tests. Furthermore, we present a formulation of the RAM required to perform FWI as well as its experimental validation.

Geophysical Model

The Marmousi velocity model, created by the Institut Françãis du Pétrole (IFP) in 1988 (Figure 4a), is used to test the performance of FWI. Particularly, we use a section of this model of 5.25 km × 1.7 km (a grid of 210×68 points with a spatial resolution of 25 m) as depicted in Figure 4b.

The observed and modeled data were produced using 100 sources located at a depth of 75 m. The first source

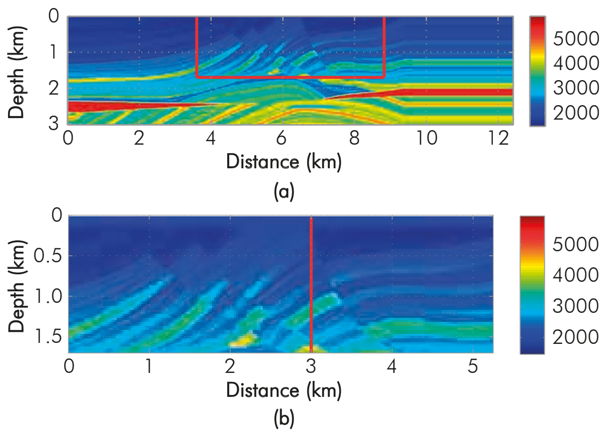


Figure 4. (a) Original Marmousi velocity model. (b) Marmousi trimmed section. The bar colors represents the velocities in m/s.

was placed at 625 m and the final source at 4625 m. The source distribution was computed using

$$S_x = S_R \cdot \left\lfloor a + \frac{(b-a)}{(n-1)} \cdot k \right\rceil \quad k=\{0, 1, 2, \cdots, n-1\} \tag{14}$$

where a is the grid position of the first source (a = 25), b the grid position of the last source (b = 185), n the number of sources (n = 100), S_R the spatial resolution (S_R = 25 m) and $\lfloor\,\rceil$ represents the operator that takes the nearest integer. The receivers were placed in line on the surface every 25 m, from the position 525 m to the position 4750 m totaling 170. Each receiver recorded 3.5 s at a time step of 4 ms and each source is a Ricker wavelet with a central frequency of 3 Hz. *Equation 4* and the Marmousi model (Figure 5a) were used to obtain the observed data. Figure 5b depicts the initial velocity model, $\mathbf{v}^0$, which was generated by applying a horizontal recursive smoothing filter to the original model. The filter was recursively applied twenty times using *Equation 15*, where $v_{i,j}^n$ represents a specific pixel in the velocity model.

$$v_{i,j}^n = \frac{1}{5} \cdot (v_{i,j-2}^n + v_{i,j-1}^n + v_{i,j}^n + v_{i,j+1}^n + v_{i,j+2}^n). \tag{15}$$

The estimated velocity model after 150 iterations is depicted in Figure 5c. At each iteration, the model is updated using *Equation 12*, with α = 10. Note in Figure 5c that the low frequency components in the resulting velocity model are correctly estimated. Figure 6 shows a 1D profile of the estimated velocity model. This 1D profile allows for an easy observation of the quality of results.

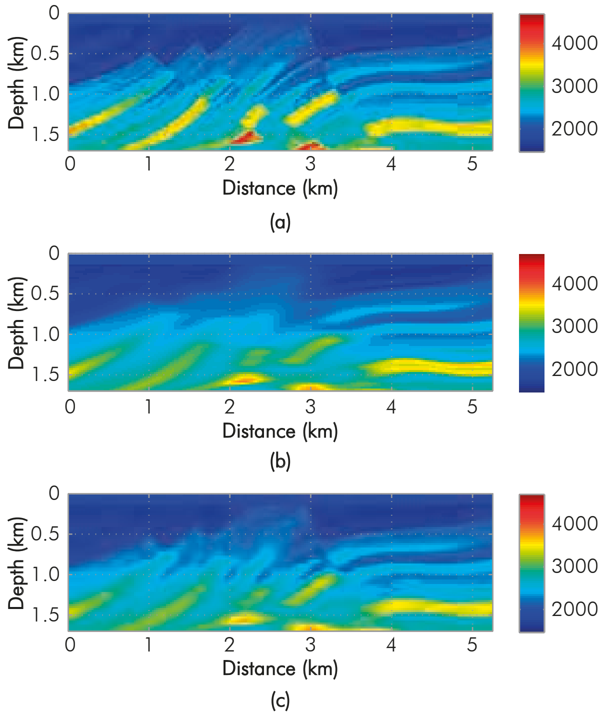


Figure 5. (a) Original Marmousi velocity model. (b) Starting velocity seismic model. (c) Estimated seismic velocity model using FWI after 150 iterations. The bar colors represents the velocities in m/s.

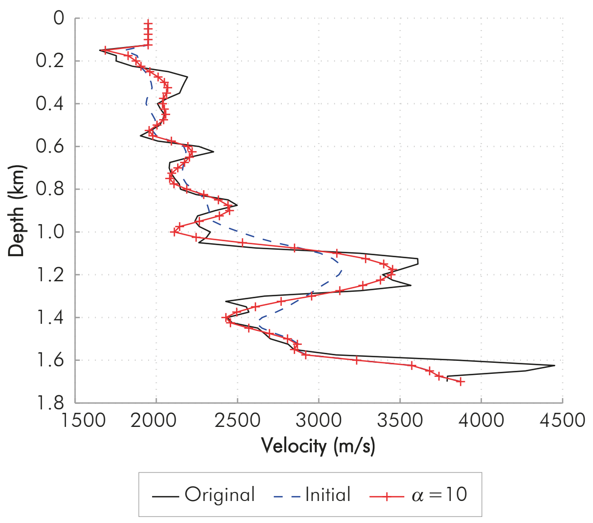


Figure 6. 1D profile at 3 km after 150 iteration of FWI using 100 shots.

Hardware Description

In this work, we present the performance measurements of the implementation of FWI in a parallel architecture based on GPU, and we compare it to the performance of its serial implementation in a CPU. The technical specifications of the CPU, and the GPU architectures are given in Tables 1, and 2, respectively. We evaluate the performance of the implementation using three main elements: running times, GPU occupancy and RAM requirements.

Table 1. Intel Core i7-4700HQ architecture.

CPU architecture	Details
Processor	Intel Core i7-4700HQ
Frequency	2.4 GHz
CPU cores	4
Cache size	6144 kB
RAM speed	1600 MHz
RAM size	12 GB

Table 2. Nvidia GeForce GTX 860M architecture.

Item	GTX 860M
Stream Multiprocessors (SMs)	5
Blocks per SM	2
Threads per Block (Max.)	1024
Threads per Block (Min.)	32
32-bits registers per Multiprocessor (Max.)	65536
Global memory	2GB
L2 cache size	256kB
Shared memory per block	6kB
Clk frequency	540MHz

Running Times

At each iteration of FWI, two wave propagations over the seismic velocity model should be computed, for each shot. If the model size increases, then the number of floating point operations increases. Additionally, each iteration also requires the computation of the gradient and the model update. We measure the execution time of 150 iterations of FWI, which are shown in Table 3. The parallel implementation on the GPU architecture is about 26.89 times faster than its serial ANSI-C implementation over the Intel Core i7 architecture.

Table 3. Execution times of FWI for 150 iterations.

Architecture	Language	Performance HH:MM:SS
Intel Core i7-4700HQ	ANSI-C	9:24:44
GTX 860M	CUDA-C	0:21:00

GPU Occupancy

The GPU occupancy metric is defined as the number of threads that are active in the GPU divided by the maximum number of threads. This value ranges from 0 to 100% and measures the efficiency of the proposed implementation. A high occupancy level means an efficient use of the available resources on the GPU. In Table 4, we show the performance of the proposed implementation on a GeForce 860M GPU, in terms of its GPU occupancy. In particular, the kernels ”Propagator”, ”Residual” and ”Gradient” have the higher use of threads in our FWI implementation. Those kernels have a 62.2, 81.5 and 88.3% of occupancy, respectively.

Table 4. GPU occupancy for the kernels in FWI for the GeForce 860M.

Kernel	Max	Min	Avg
Propagator	57.9	67.2	62.2
Gradient	64.8	90.5	88.3
CPMLx	2.2	2.2	2.2
CPMLy	1.5	1.5	1.5
Residual	80.0	84.7	81.5

RAM Requirements

The RAM required to compute FWI using the proposed implementation is given by

$$RAM\ size\ (MiB) = \frac{\beta}{1024^2 \times 8} \cdot (N_s \cdot N_t \cdot N_{trace} + N_t + 11 \cdot N_x \cdot N_z + N_x \cdot N_t + 2 \cdot N_x \cdot N_z \cdot N_t + 4 \cdot N_x + 4 \cdot N_z + N_t \cdot N_{trace}) + RD \qquad (16)$$

where N_s is the number of shots, N_t is the number of time steps, N_{trace} is the number of traces, N_x, and N_z are the spatial dimensions, β is the number of bits, and RD is the RAM required to deploy the process (for our GPU, RD = 77 MiB).

Every term inside the parenthesis of the right side of *Equation 16* represents the memory space to store the inputs or outputs of the FWI implementation. Specifically, the first term represents the observed data; the second term the wavelet used in the source; the third term represents the stencil and auxiliary variables required to solve *Equation 4*, the velocity model and the gradient; the fourth term represents the modeled data; the fifth term the volumes of the propagation and back-propagation processes; the sixth term the CPML vector for the x axis attenuation; the seventh term the CPML vector for the z axis attenuation; and finally, the eighth term represents the residuals used in the back-propagation process. For the geophysical experiment studied in this work, the numerical values of the parameters are N_s = 100, N_t = 875, N_{trace} = 170, N_x = 210, N_z = 68, and β = 32. The theoretical amount of RAM required is 230.9481 MiB. We ran various tests to validate *Equation 16*. In particular we keep the same model size and we change the number of shots used in FWI. The RAM used at each experiment and the theoretical RAM are shown in Table 5.

Table 5. Theoretical and measured RAM required by FWI for different number of shots.

Shots	Measured RAM	Theoretical RAM
5	177 MiB	177.0416 MiB
11	180 MiB	180.4463 MiB
21	186 MiB	186.1206 MiB
31	192 MiB	191.7950 MiB
100	231 MiB	230.9481 MiB

Currently, a GPU developed for high performance computing applications like NVIDIA Tesla K40 (chip GK110B) have a maximum of 12GiB of RAM, which could be a constraint to implement a full 3D acoustic FWI inside the GPU. A 3D geophysical experiment would require either more sophisticated computer architectures or implementation strategies that takes into account the data transfer from RAM to Disk or from Disk to RAM.

5. DISCUSSION

In this work we used a finite-differences approximation, in time domain, to find the numerical solution of the 2D acoustic wave equation. The approximation used in this work is of second order in time and space. We found that a second order approximation gives adequate results in the quality of the estimated synthetic velocity model, and additionally it offers low computational cost. Further analysis can be done on evaluating the use of a second order approximation in real geophysical problems. An improvement in the resolution of the velocity models can be obtained with higher order approximations at the expense of higher computational cost.

The use of parallel programming in GPUs accelerates the computation of FWI in comparison to a CPU serial implementation, since this algorithm is highly parallelable. The performance comparison between our GPU implementation and a CPU parallel implementation remains as on-going research work. One of the main concerns, in the GPU implementation, arises in terms of RAM requirements. For the geophysical experiment studied in this paper, the RAM requirements were satisfied. Nevertheless, real geophysical surveys can require RAM resources larger than 12 GiB. For example, in a 3D FWI, the RAM requirements of our implementation would not be satisfied by current GPU technologies, and FWI implementation should take into account the data transfer between GPU and CPU memories.

Finally, synchronization problems might occur when FWI is computed using parallel architectures such as GPUs. It is critical to implement the algorithm taking into account the synchronization of all the processes running inside the GPU. In the proposed GPU implementation, all points of the future pressure field layer are computed in parallel. Therefore, it is critical to guarantee that present and past pressure field layers are computed before computing the future layer, otherwise an error will be propagated throughout all the layers.

6. CONCLUSIONS

- In this paper, we presented a GPU parallel implementation of the 2D acoustic full waveform inversion. The proposed implementation uses finite-differences in time domain to find the solution of the isotropic and acoustic wave equation. Also, we used the convolutional perfect matched layer method to compute the propagated field at the non-natural boundaries. The algorithm uses the adjoint state method to update the velocity models iteratively.

- The proposed parallel implementation was tested for the GPU GTX 860M in terms of running times, RAM requirements and GPU occupancy. Running times were compared to those obtained using serial programming on an Intel Core i7-4700HQ architecture. The high parallelism level of FWI algorithm makes GPU implementation about 26.89 times faster than our sequential implementation. The RAM requirements proposed by *Equation 16* were experimentally validated, showing the impact of storing in RAM all the observed data and both wave fields in a 2D FWI. The GPU occupancy presented in Table 4 reveals that none of the kernels uses 100 percent of the resources. This is intrinsic to the propagation process because some kernels are active inside the CPML area and others are active outside the CPML area, but all of them have the same size of the geophysical model ($N_x \cdot N_z$).

- In the geophysical experiment the major constraint of the proposed implementation is the memory capacity. Only 230.94 MiB was required for our 2D FWI implementation, for all shot gathers; however a 3D implementation can easily require more than 12 GiB of RAM (the current capacity of a GPU Tesla K40). In that case, a different strategy for implementing FWI should be tested where only one field is stored in RAM, and the other is constantly re-computed. This strategy would use less RAM allowing a 3D implementation.

ACKNOWLEDGEMENTS

This work is supported by Colombian Oil Company *Ecopetrol S.A.* and *Colciencias* as a part of the research project grant No. 0266-2013.

REFERENCES

Alford, R., Kelly, K. & Boore, D. M. (1974). Accuracy of finite-difference modeling of the acoustic wave equation. *Geophysics*, 39(6), 834-842.

Berenger, J. P. (1994). A perfectly matched layer for the absorption of electromagnetic waves. *J. Comput. Phy.*, 114(2), 185-200.

Bunks, C., Saleck, F. M., Zaleski, S. & Chavent, G. (1995). Multiscale seismic waveform inversion. *Geophysics*, 60(5), 1457-1473.

Cao, D. & Liao, W. (2014). An adjoint-based hybrid computational method for crosswell seismic inversion. *Comput. Sci. Eng.*, 16(6), 60-67.

Collino, F. & Tsogka, C. (2001). Application of the perfectly matched absorbing layer model to the linear elastodynamic problem in anisotropic heterogeneous media. *Geophysics*, 66(1), 294-307.

Etienne, V., Tonellot, T., Thierry, P., Berthoumieux, V. & Andreolli, C. (2014). Speeding-up FWI by one order of magnitude. *EAGE Workshop on High Performance Computing for Upstream.* Chania, Crete.

Hu, W., Abubakar, A. & Habashy, T. M. (2007). Application of the nearly perfectly matched layer in acoustic wave modeling. *Geophysics*, 72(5), 169-175.

Kim, Y., Shin, C. & Calandra, H. (2012). 3D hybrid waveform inversion with GPU devices. *SEG Annual Meeting*. Las Vegas, Nevada. SEG-2012-0138.

Lailly, P. (1983). The seismic inverse problem as a sequence of before stack migrations. *Conference on Inverse Scattering: Theory and Application*. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Mao, J., Wu, R.S. & Wang, B. (2012). Multiscale full waveform inversion using GPU. *SEG Annual Meeting*. Las Vegas, Nevada, SEG-2012-2575.

Pasalic, D. & McGarry, R. (2010). Convolutional perfectly matched layer for isotropic and anisotropic acoustic wave equations. *SEG Annual Meeting*. Denver, Colorado, SEG-2010-2925.

Plessix, R. E. (2006). A review of the adjoint-state method for computing the gradient of a functional with geophysical applications. *Geophys. J. Int.*, 167(2), 495-503.

Tarantola, A. (1984). Inversion of seismic reflection data in the acoustic approximation. *Geophysics*, 49(8), 1259-1266.

Tarantola, A. (1987). *Inverse problem theory: Methods for data fitting and model parameter estimation*. Elsevier.

Thierry, P., Donno, D. & Noble, M. (2014). Fast 2D FWI on a multi and many-cores workstation. *EGU General Assembly*, 16: Vienna, Austria.

Versteeg, R. & Grau, G. (1991). The Marmousi experience: *Proceedings of the 1990 EAEG Workshop on practical aspects of seismic data inversion*.

Vigh, D., Cheng, X., Jiao, K., Sun, D. & Kapoor, J. (2014). Multiparameter TTI full waveform inversion on long-offset broadband acquisition: A case of study. *SEG Annual Meeting.* Denver, Colorado, SEG-2014-0530.

Virieux, J. & Operto, S. (2009). An overview of full-waveform inversion in exploration geophysics. *Geophysics,* 74(6), 1-26.

Wang, B., Gao, J., Zhang, H. & Zhao,W. (2011). CUDA-based acceleration of full waveform inversion on GPU. *SEG Annual Meeting*. San Antonio, Texas, SEG-2011-2528.

Weiss, R. M. & Shragge, J. (2013). Solving 3D anisotropic elastic wave equations on parallel GPU devices. *Geophysics*, 78(2), 7-15.

Yang, P., Gao, J. & Wang, B. (2015). A graphics processing unit implementation of time-domain full-waveform inversion. *Geophysics*, 80(3), 31-39.

Zhang, M., Sui, Z., Wang, H., Ren, H., Wang, Y. & Meng, X. (2014). An approach of full waveform inversion on GPU clusters and its application on land datasets. *76th EAGE Conference and Exhibition*. Amsterdam.

AUTHORS

Sergio-Alberto Abreo-Carrillo
Affiliation: *Universidad Industrial de Santander*
Electronics Engineer, *Universidad Distrital Francisco José de Caldas*
M. Eng. in Electronics Engineering, *Universidad Industrial de Santander*
Ph. D. Student in Electronics Engineering, *Universidad Industrial de Santander*
e-mail: abreosergio@gmail.com

Ana-Beatriz Ramírez-Silva
Affiliation: *Universidad Industrial de Santander*
Electronics Engineer, *Universidad Industrial de Santander*
M. Sc. in Electronics Engineering, *Recinto Universitario de Mayaguez*
Ph. D. in Electrical Engineering, *University of Delaware*
e-mail: anaberam@uis.edu.co

Oscar Reyes
Affiliation: *Universidad Industrial de Santander*
Electronics Engineer, *Universidad Industrial de Santander*
M. Eng. in Electronics Engineering, *Universidad Industrial de Santander*
Dr. -Ing. in Computer Science, *Hamburg University of Technology*
e-mail: omreyes@uis.edu.co

David-Leonardo Abreo-Carrillo
Affiliation: *Universidad Industrial de Santander*
Electronics Engineer, *Universidad Industrial de Santander*
M. Eng. Student in Electronics Engineering, *Universidad Industrial de Santander*
e-mail: david.abreo@gmail.com

Herling González-Alvarez
Affiliation: *Ecopetrol S.A. - Instituto Colombiano del Petróleo (ICP)*
Physicist, *Universidad Industrial de Santander*
M. Sc. in Physics, *Universidad Industrial de Santander*
e-mail: herling.gonzalez@ecopetrol.com.co