IMPLEMENTATION STRATEGIES OF THE SEISMIC FULL WAVEFORM INVERSION

Reynaldo F. Noriega †*, Ana B. Ramirez* †*, Sergio A. Abreo*†*, and Gonzalo R. Arce*‡.

†Industrial University of Santander, Bucaramanga, Colombia.
‡ University of Delaware, Newark, DE.

ABSTRACT

Full waveform inversion (FWI) is a state-of-the-art method that has been used to estimate parameters of the Earth's subsurface. One of the main drawbacks of the FWI method is its high computational complexity in terms of both time and memory. This occurs because the inversion method is based on the computation of a gradient function that requires the forward (and backward) wave propagation of sources (and residuals) through the subsurface medium. Nowdays, seismic surveys are large scale problems and therefore the computation and storage in RAM memory of both wavefields is not feasible. Therefore, different strategies for the FWI method should be used. In this paper, we use two different implementation strategies that avoid allocating the full wavefields in RAM memory. Instead, the wavefields are re-computed while at the same time the gradient function is obtained. The re-computation of the wavefields remains possible from a practical point of view since we use parallel architectures (GPUs). We show that the estimated velocity models obtained with all the strategies are similar. We also show that the RAM consumption decreases up to 80% for the proposed strategies in comparison with the strategy that requires storing the full wavefields.

Index Terms— Seismic full waveform inversion, CPML, GPUs, FDTD.

1. INTRODUCTION

Full Waveform Inversion (FWI) is a state-of-the-art method used to estimate parameters of the Earth's subsurface from seismic data acquired at the surface [1, 2]. Currently, this inversion method has gain a lot of attention due to its capability of finding velocity models of high resolution. The FWI method consists on finding the model that minimizes the ℓ_2-error norm between the observed and modeled seismic traces. The modeled data is generated by using an initial guess of the velocity model and a wave equation model. A good starting model is required to find an adequate estimate, otherwise the solution is a local minima. Additionally, a strategy to avoid artificial reflections due to non-natural boundaries should be considered when using a wave equation model. Among the traditional boundary conditions, the convolutional perfectly matched layer (CPML) method absorbs adequately the non-natural reflections in the solution of the wave equation [3].

The FWI method iteratively estimates the velocity model of the subsurface by using a gradient descent approach. The computation of the gradient demands a high cost of time and storage resources because it requires computing the forward wavefield of the sources and the backward wavefield of the residuals, through the subsurface medium [4]. In particular, we exploit the parallelism level of computing the wavefields using finite differences time domain (FDTD), such that every spatial point at a single time snapshot can be obtained at once in a graphics processing unit(GPU) architecture [5]. The parallelism level of the FDTD also favors the use of implementation strategies that do not need to store large amounts of data, because the wavefields can be re-computed while at the same time the gradient function is obtained. In this paper, we compare different FWI implementation strategies that manage the RAM memory consumption and does not increase the computational time considerably. One strategy requires storing only one wavefield. A different strategy re-computes wavefields at each iteration [6], so that much less RAM memory is required. Clearly, the implementation strategies would need more execution time than if both wavefields are allocated in memory.

2. THEORETICAL BACKGROUND

Seismic FWI is a non-linear inversion method that aims at estimating subsurface properties, such as the velocity model, from seismic data recorded at the surface. The method estimates the set of parameters $\mathbf{m}$ that minimizes the ℓ_2-norm of the error between the observed and modeled seismic data. The inversion method is given by the following optimization problem [2]

$$\hat{\mathbf{m}} = \arg\min_{\mathbf{m}} \Phi(\mathbf{m}) = \arg\min_{\mathbf{m}} \frac{1}{2} \| \, d_{obs} - d_{mod}(\mathbf{m}) \, \|_2^2, \quad (1)$$

where d_{obs} and $d_{mod}(\mathbf{m})$ are the acquired and modeled seismic data, respectively. The modeled seismic data $d_{mod}(\mathbf{m})$ is the solution at the surface position of the two-way wave equation given by

$$\frac{1}{m^2(x,z)} \frac{\partial^2 \rho}{\partial t^2} = \frac{\partial^2 \rho}{\partial x^2} + \frac{\partial^2 \rho}{\partial z^2} + S(x,z,t), \quad (2)$$

where $S(x,z,t)$ is the source wavelet that depends on the spatial and temporal variables.
A strategy to find the solution of the problem stated in (1) is using the gradient descent method, such that an update for the parameters at iteration $k+1$ is given by

$$\mathbf{m}_{k+1} = \mathbf{m}_k + \alpha \cdot \mathbf{g}_k, \tag{3}$$

where $\mathbf{m}_k$ and $\mathbf{g}_k$ are the velocity model at the k^{th} iteration and the gradient, respectively. α is the step size that controls the update of the model. The gradient $\mathbf{g}_k$ uses the forward and backward propagation wavefields of the source and residual data, respectively. According to [7], the gradient $\mathbf{g}_k$ can be obtained using the first order adjoint state method, and it is given by

$$g_k(x,z) = -\frac{2}{m_k^3(x,z)} \sum_s \int_0^T q_s(x,z,T-t) \frac{\partial^2 p_s(x,z,t)}{\partial t^2} dt, \tag{4}$$

where $q_s(x,z,T-t)$ is the backward propagated field in time when the source is the residual between observed and modeled data, $p_s(x,z,t)$ is the forward propagated field of the source s, and T is the total wavefield propagation time.
The modeling of the wave propagation using the acoustic wave equation uses absorbing boundary conditions in order to eliminate the reflections given by non-natural boundaries. In order to reduce artificial reflections within the area of interest, we used the convoluted perfectly matched layer (CPML) method. The modified wave equation that includes CPML at the non-natural boundaries is given by

$$\frac{1}{m^2(x,z)} \frac{\partial^2 \rho}{\partial t^2} = \frac{\partial^2 \rho}{\partial x^2} + \frac{\partial^2 \rho}{\partial z^2} + S(x,z,t) + \frac{\partial^2 \psi(x,z)}{\partial t^2} + \xi(x,z), \tag{5}$$

where $\psi(x,z)$, $\xi(x,z)$ are auxiliar fields such that the wavefield at the boundaries decays. A complete description of how to select $\psi(x,z)$ and $\xi(x,z)$ can be found in [8].

2.1. Full waveform inversion algorithm

The traditional implementation for the FWI algorithm is given in form of a pseudo-code in Algorithm 1. Note that in line 22 of Algorithm 1, that both volumes should be stored in RAM memory, the forward and backward propagated wavefields. Furthermore, the execution time of the FWI depends on the number of sources, total propagation time, and time and spatial discretization.

Algorithm 1 Full Waveform Inversion

1: $FWI(m, Tobs, S, inf, \alpha)$ ▷ FWI inputs
2: m ▷ Starting velocity model
3: $Tobs$ ▷ Acquired data
4: S ▷ Wavelet source
5: inf ▷ Location of each shot
6: α ▷ Alpha value
7: **for** $i \leftarrow 1, iG$ **do** ▷ iG, Number of FWI iterations
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: $\frac{1}{m(x,z)^2}\frac{\partial^2 \rho}{\partial t^2} = \beta + \frac{\partial^2 \rho}{\partial x^2} + \frac{\partial^2 \rho}{\partial z^2} + S(sj,t)$
13: $Mod(x,t) \leftarrow \rho(x,0,t)$
14: **end for**
15: $Obs_{dev} \leftarrow Obs_{j,host}$ ▷ Observed data from host to device
16: $Residual(x,t) \leftarrow Tmod(x,t) - Tobs_{dev}(x,t)$
17: $norm \leftarrow \|Residual(x,t)\|_2^2$ ▷ (L2 Norm)^2
18: $acum \leftarrow \frac{1}{2} norm + acum$
19: **for** $t \leftarrow 1, Nt$ **do**
20: $\frac{1}{m(x,z)^2}\frac{\partial^2 \lambda}{\partial t^2} = \beta + \frac{\partial^2 \lambda}{\partial x^2} + \frac{\partial^2 \lambda}{\partial z^2} + Residual(x, Nt-t)$
21: **end for** ▷ Gradient
22: $g(x,z) = g(x,z) - \frac{2}{(m^k(\mathbf{x},\mathbf{z}))^3} \int_0^T \lambda(x,z,T-t) \frac{\partial^2 \rho(x,z,t)}{\partial t^2} dt$
23: **end for**
24: $\phi(i) \leftarrow acum$
25: $m(x,z) \leftarrow m(x,z) + \alpha \cdot g(x,z)$ ▷ Model update
26: **end for**
27: $m_{end} \leftarrow m(x,z)$ ▷ Final velocity model from device to host.
28: **return** ϕ, m_{end}

3. FWI STRATEGIES AND COMPUTATIONAL PERFORMANCE ON GPUS

The principal advantage of using graphical processing units (GPUs) for the FWI implementation is that we can compute every snapshot of the wave propagation in parallel. Thus, at the same time, the pressure wavefield at all spatial positions of one snapshot are computed. However, as mentioned before, the implementation of a FWI algorithm has an expensive RAM consumption, because to the backward and forward wavefield volumes need to be stored. Those volumes require up to between 80% and 90% of the memory used by the implementation. For that reason, the use of GPU architectures has been limited in real problems and a feasible solution is to find different implementation strategies. In the following, three different implementation strategies are compared.

3.1. Strategy I: Dual volume allocation

The first strategy is the common FWI implementation, which allocates GPU RAM for all shot-gathers, velocity models, gradient and the backward and forward propagation volumes [4]. A GPU function performs the propagation and stores the backward and forward wavefields to then compute the gradient matrix by first making a dot product of the wavefields, and then summing the resulting volume in the time direction. The main steps of the first implementation strategy is given in Algorithm 2.

3.2. Strategy II: Single volume allocation

The second FWI implementation strategy consist on allocating GPU RAM for all matrices, shot-gathers, velocity model and a single propagation volume. A GPU kernel function computes the forward wavefield and stores it. A second

Algorithm 2 : Dual volume allocation

1: **for** $t \leftarrow 1, Nt$ **do** ▷ Nt, Number of time steps
2: $\frac{1}{m(x,z)^2}\frac{\partial^2\rho}{\partial t^2} = \beta + \frac{\partial^2\rho}{\partial x^2} + \frac{\partial^2\rho}{\partial z^2} + S(sj,t)$
3: **end for**
4: **for** $t \leftarrow 1, Nt$ **do**
5: $\frac{1}{m(x,z)^2}\frac{\partial^2\lambda}{\partial t^2} = \beta + \frac{\partial^2\lambda}{\partial x^2} + \frac{\partial^2\lambda}{\partial z^2} + Residual(x, Nt-t)$
6: **end for**
7: $g(x,z) = g(x,z) - \frac{2}{(m^k(\mathbf{x},\mathbf{z}))^3}\int_0^T \lambda(x,z,T-t)\frac{\partial^2\rho(x,z,t)}{\partial t^2}dt$ ▷ Gradient

GPU kernel computes the backward propagated wavefield and overwrites the allocated memory with the dot product between the computed snapshot of the backward wavefield and the stored forward wavefield. Finally, a third GPU kernel adds all the snapshots in order to obtain the gradient matrix. The main steps of the second implementation strategy is given in Algorithm 3.

3.3. Strategy III: No volume allocation

The third FWI implementation strategy requires no GPU RAM allocation for the wavefields volumes. Instead, we first compute the forward propagation in order to obtain modeled data and compute the residual traces. Then, backward propagation is re-computed using only the two snapshots $\rho(x,z,1)$ and $\rho(x,z,2)$, together with the CPML boundaries for each time step. A GPU kernel re-computes the backward propagation from the stored data while computing the forward propagation wavefield once again. The same GPU kernels computes gradient matrix by multiplying the forward snapshot and the reconstructed backward snapshot, and it stores the sum of all resulting matrices. The main steps of the third implementation strategy is given in Algorithm 4.

4. NUMERICAL EXPERIMENTS AND RESULTS

The three strategies were evaluated using a finite-difference time domain approximation of 8^{th}-order in space, and 2^{nd}-order in time. The true velocity model was used to generate the observed data, and a initial velocity model guess is obtained by applying a low-pass filter to the true velocity model 20 times horizontally. The parameters of the synthetic model are $Nx = 497$, $Nz = 121$, $\Delta_x = \Delta_z = 0.025$ Km, 457 receivers, 20 grid points of CPML boundaries, $\Delta_t = 1$ ms and the total propagation time was $T = 3.5$ s. A $ricker$ wavelet was used as source with frequencies of 5, 10 and 20 Hz, and constant step in the descent direction of $\alpha = 9.0$. The estimated velocity models for all strategies are depicted in Fig. 1.

We also present in Table 1 a comparison between the theoretical proposed GPU-RAM consumption given by equations (6), (7), and (8) for strategies I, II and III, respectively; with the measured GPU-RAM. In equations (6), (7), and (8), RD is the RAM used by the GPU for launching the process,

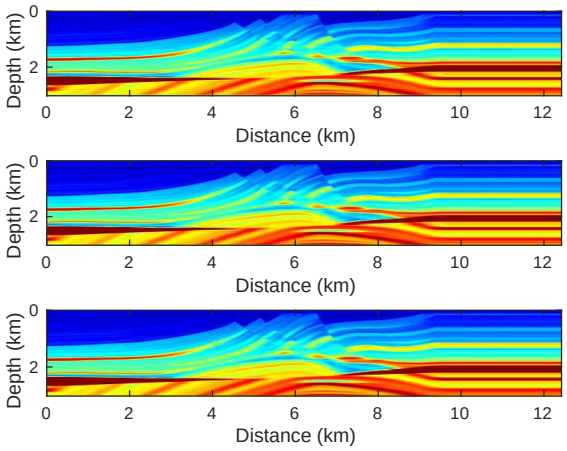

Fig. 1. Final velocity model obtained with: Top: strategy I. Middle: strategy II. Bottom: strategy III.

Algorithm 3 : Single volume allocation

1: **for** $t \leftarrow 1, Nt$ **do** ▷ Nt, Number of time steps
2: $\frac{1}{m(x,z)^2}\frac{\partial^2\rho}{\partial t^2} = \frac{\partial^2\rho}{\partial x^2} + \frac{\partial^2\rho}{\partial z^2} + S(sj,t)$
3: **end for**
4: **for** $t \leftarrow 1, Nt$ **do**
5: $\frac{1}{m(x,z)^2}\frac{\partial^2\lambda}{\partial t^2} = \beta + \frac{\partial^2\lambda}{\partial x^2} + \frac{\partial^2\lambda}{\partial z^2} + Residual(x, Nt-t)$
6: $\frac{\partial^2\rho(x,z,Nt-t)}{\partial t^2} = \beta + \frac{\partial^2\rho(x,z,Nt-t)}{\partial t^2} * \lambda(x,z,Nt-t)$
7: **end for**
8: $g(x,z) = g(x,z) - \frac{2}{(m^k(\mathbf{x},\mathbf{z}))^3}\int_0^T \frac{\partial^2\rho(x,z,t)}{\partial t^2}dt$ ▷ Gradient

and it is constant for all the strategies; $\sigma = 4$ is the size of a float variable, and St is the size of the spatial stencil used for the finite-differences approximation. Figure 2 shows the GPU-RAM consumption of strategy I, II and III, relative to strategy I, as a function of the number of shots. Note that as the number of shots increases, the reduction in memory consumption obtained with strategy II and III decreases. This

Algorithm 4 : No volume allocation

1: **for** $t \leftarrow 1, Nt$ **do** ▷ Nt, Number of time steps
2: $\frac{1}{m(x,z)^2}\frac{\partial^2\rho}{\partial t^2} = \beta + \frac{\partial^2\rho}{\partial x^2} + \frac{\partial^2\rho}{\partial z^2} + S(sj,t)$
3: **end for**
4: **for** $t \leftarrow Nt, 1$ **do** ▷ Nt, Number of time steps
5: $\frac{1}{m(x,z)^2}\frac{\partial^2\lambda}{\partial t^2} = \beta + \frac{\partial^2\lambda}{\partial x^2} + \frac{\partial^2\lambda}{\partial z^2} + lambda(sj,t)$
6: **end for**
7: $save \leftarrow \rho(x,z,1)\ and\ \rho(x,z,2)$ ▷ Initial conditions for next propagation
8: **for** $t \leftarrow 1, Nt$ **do**
9: $\frac{1}{m(x,z)^2}\frac{\partial^2\lambda}{\partial t^2} = \frac{\partial^2\lambda}{\partial x^2} + \frac{\partial^2\lambda}{\partial z^2} - Residual(x,t)$
10: $\frac{1}{m(x,z)^2}\frac{\partial^2\rho}{\partial t^2} = \beta + \frac{\partial^2\rho}{\partial x^2} + \frac{\partial^2\rho}{\partial z^2} + S(sj,t)$
11: $g(x,z) = g(x,z) - \frac{2}{(m^k(bfx,\mathbf{z}))^3}\frac{\partial^2\rho(x,z,t)}{\partial t^2}\lambda(x,z,t)$ ▷ Gradient
12: **end for**

occurs since the memory required to allocate the backward and forward wavefields is less significant than the memory required to allocate the observed data(using several number of sources).
Finally, in order to guarantee that the estimated velocity models are the same for all the strategies, we compute the quadratic error between the true and the estimated model for the three strategies for 1, 51 and 171 shots. This is shown in Table 2. Note that all strategies have the same error norm when 1 shot is used. However, when more than one shot are used, the error norm increases in strategies II and III. This occurs because performing the operations in different order inside the GPU gives different result due to rounding approximations.

5. CONCLUSIONS

In this work, we compare three different implementation strategies for the FWI algorithm, which can be used to decrease the RAM consumption on CUDA heterogeneous GPU architectures. A small difference in the resulting velocity model is found in the different strategies, which is generated due to the different order of the operators used in the implementations. The storage consumption is reduced significantly for small number of shots, which allows the use of the proposed strategies for the estimation of velocity model is real seismic applications.

6. ACKNOWLEDGMENTS

This work is partially supported by Colombian Oil Company ECOPETROL and COLCIENCIAS as part of the research project grant No. 0266-2013.

$$\text{RAM\_SI (MiB)} = RD + \sigma * (20 * Nx * Nz + 3 * Nx * Nt + 2 * Nx * Nz * Nt + Nx * Nt * Ns + 2 * Nx + 2 * Nz)/(2^{20}). \quad (6)$$

$$\text{RAM\_SII(MiB)} = RD + \sigma * (20 * Nx * Nz + 3 * Nx * Nt + Nx * Nz * Nt + Nx * Nt * Ns + 2 * Nx + 2 * Nz)/(2^{20}). \quad (7)$$

$$\text{RAM\_SIII(MiB)} = RD + \sigma * (20 * Nx * Nz + 4 * Nx * Nt + Nx * Nt * Ns + 2 * Nx + 2 * Nz + 2 * St * Nz * Nt + 2 * St * Nx * Nt)/(2^{20}). \quad (8)$$

No. of shots	Strategy	Measur.[MiB]	Theor.[MiB]
1	I	1724	1723. 85
	II	921	920.93
	III	362	361.55
51	I	2056	2055.63
	II	1253	1252.72
	III	693	693.34

Table 1. Measured vs. theoretical RAM consumption for all strategies for different number of shots.

Strat. / No.sources	1	51	171
I	30903.53	28260.37	26536.13
II	30903.53	28261.31	26536.13
III	30903.53	28264.20	26544.02

Table 2. ℓ_2-error norm between true and estimated model for all strategies for different number of sources.

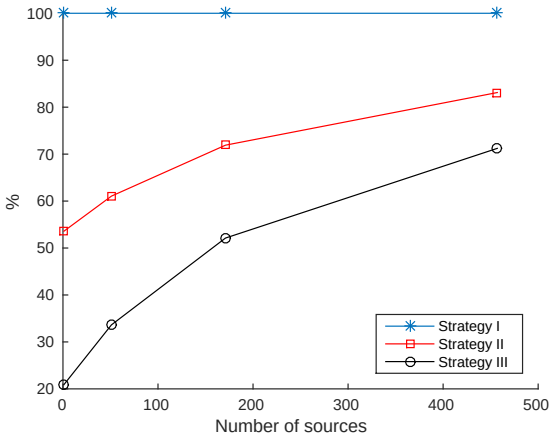

Fig. 2. Percentage of RAM consumption for strategies I, II and III, relative to strategy I.

7. REFERENCES

[1] A. Tarantola, *Inverse Problem Theory and Methods for Model Parameter Estimation*, Addison-Wesley, 1989.

[2] J. Virieux and S. Operto, "An overview of full-waveform inversion in exploration geophysics," *Geophysics*, vol. 74, no. 3, pp. no. 6, 2009.

[3] D. Pasalic, *Convolutional perfectly matched layer for isotropic and anisotropic acoustic wave equations*, Addison-Wesley, 2010.

[4] S.-A. Abreo, A. Ramirez, O. Reyes, D.-L. Abreo, and H. Gonzalez, "Practical implementation of acoustic full waveform inversion on graphical processing units," *CTYF*, vol. 6, no. 2, pp. 5–16, 2015.

[5] X.-D. Tang, H. Zhang, and Hong Liu, "Frequency-space domain acoustic modeling based on an average-derivative and gpu implementation," *Chinese Journal of Geophysics*, vol. 58, 2015.

[6] Pengliang Yang, Jinghuai Gao, and OBaoli Wang, "A graphics processing unit implementationof time-domain full-waveform inversion," *Geophysics*, vol. 80, no. 3, pp. 4–5, 2015.

[7] R Plessix., "A review of the adjoint-state method for computing the gradient of a functional with geophysical applications," *Geophysics*, vol. 167, pp. 495–503, 2006.

[8] F. Collino and C. Tsogka, *Application of the perfectly matched absorbing layer model to the linear elasto-dynamic problem in anisotropic heterogeneous media*, Addison-Wesley, 2004.