INTRODUCTION

The RTM implementation using the random boundary condition (RBC) was proposed by **?** as an alternative to checkpointing (e.g. **?**), to avoid the storage of the source wavefield propagation history at the cost of an additional time-reverse propagation to re-generate it. **?** further adapted the technique to cope with low-wavenumber limitations. This improvement is particularly beneficial to the proper computation of the FWI gradient (**?**). More recently, **?** adapted the method to elastic RTM.

Besides this "traditional" implementation of the RBC in RTM, I also adapted them for the WEMVA operator, which is one of the computationally expensive operators needed to implement JIRB. To my knowledge, RBC had not been used before in WEMVA. For this reason, this chapter is dedicated to explain how it works.

BRIEF REVIEW OF THE RANDOM BOUNDARY CONDITION ON RTM

When imaging using RTM, we perform the propagation of the wavefields in opposite time directions, i.e., the source wavefield's history is constructed by forward-time propagation of the injected source wavefield. In contrast, we construct the receiver wavefield by reverse-time propagation of the injected field data. For such reason, one of the wavefields (typically the source wavefield) remains either in RAM or physical memory during the propagation of the other wavefield, of which merely two history frames are kept in memory to perform the correlation imaging condition (**?**) on the fly. Even if storing only one of the wavefields, we still need to account for thousands of time frames that in 3D represent seismic volumes of considerable size, not to talk about the I/O latency that comes to the detriment of computational performance.

Different strategies can be implemented to reduce storage. **?** proposed the use of checkpoints. This strategy consists on saving a determined number of time frames for the re-computation of the wavefields from optimally chosen times. This method works well in practice, but it still requires a non-trivial selection of the optimum checkpoints (**?**).

Another strategy proposed by **?** is saving the boundaries of the source wavefield in a rectangular halo beyond the imaging space, and re-injecting them to perform a reverse-time propagation of the source wavefield as performing the usual reverse-time receiver wavefield propagation, while multiplying corresponding time frames to build the image. This strategy allows a re-computation similar to checkpoints, with better applicability for large volumes (**?**).

Both methods - checkpointing and saving the boundaries - require I/O work to some extent. For such reason, **?** proposed the use of the RBC to recreate the source

wavefield in reverse-time propagation. This operation only requires keeping two time frames in RAM for both the source wavefield and the receiver wavefield, thereby constituting a tremendous saving in storage. Even though the RBC requires the re-computation of the source wavefield, it avoids excessive I/O operations.

Figure 1 schematically shows the RBC implementation in comparison to the conventional RTM computation.

The upper part of Figure 1 illustrates the stages of conventional RTM implementation were the source wavefield is forward-time propagated and its whole history of n_t samples (orange frames) is stored (a). Next, the receiver wavefield (blue frames) is reverse-time propagated (b-g) and cross-correlated with the receiver wavefield on the fly. Note that only two temporal frames of the receiver wavefield need be kept in memory. This strategy works well for small 2D datasets. However, for large 2D datasets or 3D datasets, the source wavefield needs to be stored in disk, and their corresponding time frames have to be accessed as needed during the cross-correlation.

The lower part of Figure 1 illustrates the stages of RTM using the RBC. The source wavefield is forward-time propagated as before, but only two temporal frames of its history are kept in memory (a). Then it trespasses the limit reaching sample $n_t + 1$. Next, the source wavefield is reverse-time propagated simultaneously with the receiver wavefield and cross-correlated on the fly (b-g). Note that only a total of four temporal frames are kept in memory, thereby wavefield storage is no longer needed.

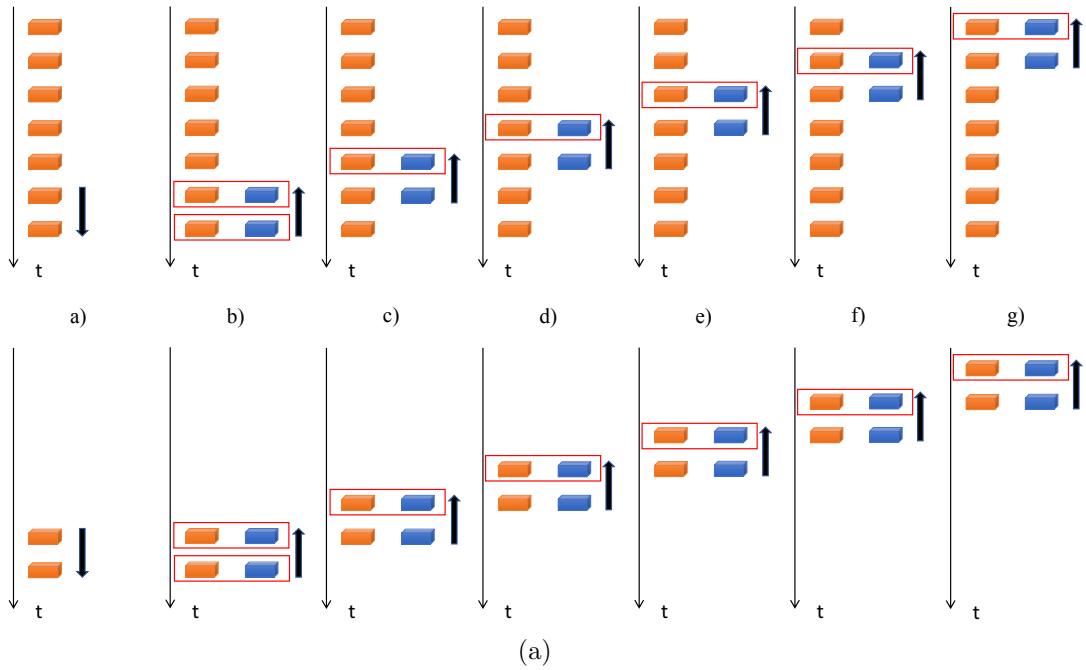

Figure 1: Comparison of conventional RTM implementation (top) and using the random boundary condition (bottom). [**NR**]

APPLICATION OF THE RANDOM BOUNDARY CONDITION TO WEMVA

The WEMVA operator has two components, namely the source-side component and the receiver-side component (**??**). Each component consists of two operations. For the source-side component, one propagates forward in time the source wavefield, then scatters this wavefield against the perturbation in the background model (forward operator) or the perturbation in the image (adjoint operator) and propagates the scattered wavefield. Finally, one propagates backward in time the receiver wavefield and cross-correlates it with the scattered source wavefield. We can use a similar argument for the computation of the receiver-side WEMVA component.

Figure 2 illustrates the different stages of the forward WEMVA operator. The WEMVA scattering stage is similar to Born scattering, whereas the WEMVA cross-correlation of the background wavefields with the scattered wavefields is similar to the RTM imaging condition. It is during the latter stage that we can apply the RBC.

We can implement the WEMVA operator as follows:

- Forward propagation of the source wavefield in the background slowness field.
- Backward propagation of the receiver wavefield in the background slowness field.
- Scattering of the source wavefield upon the perturbation in the image or the perturbation in the background model.
- Scattering of the receiver wavefield upon the perturbation in the image or the perturbation in the background model.
- Zero-lag time cross-correlation of the source wavefield and scattered receiver wavefield (receiver side).
- Zero-lag time cross-correlation of the receiver wavefield and scattered source wavefield (source side).
- Addition of source and receiver sides of WEMVA.

Next, I show the implementation of the above steps when we store the background source and receiver wavefields, then when we store only one background wavefield, and finally, when applying RBC that we do not have to store anything.

Implementation of the WEMVA operator storing the background wavefields

If we can afford to store the propagated wavefields, then we can execute the WEMVA process as described in Algorithm 1.

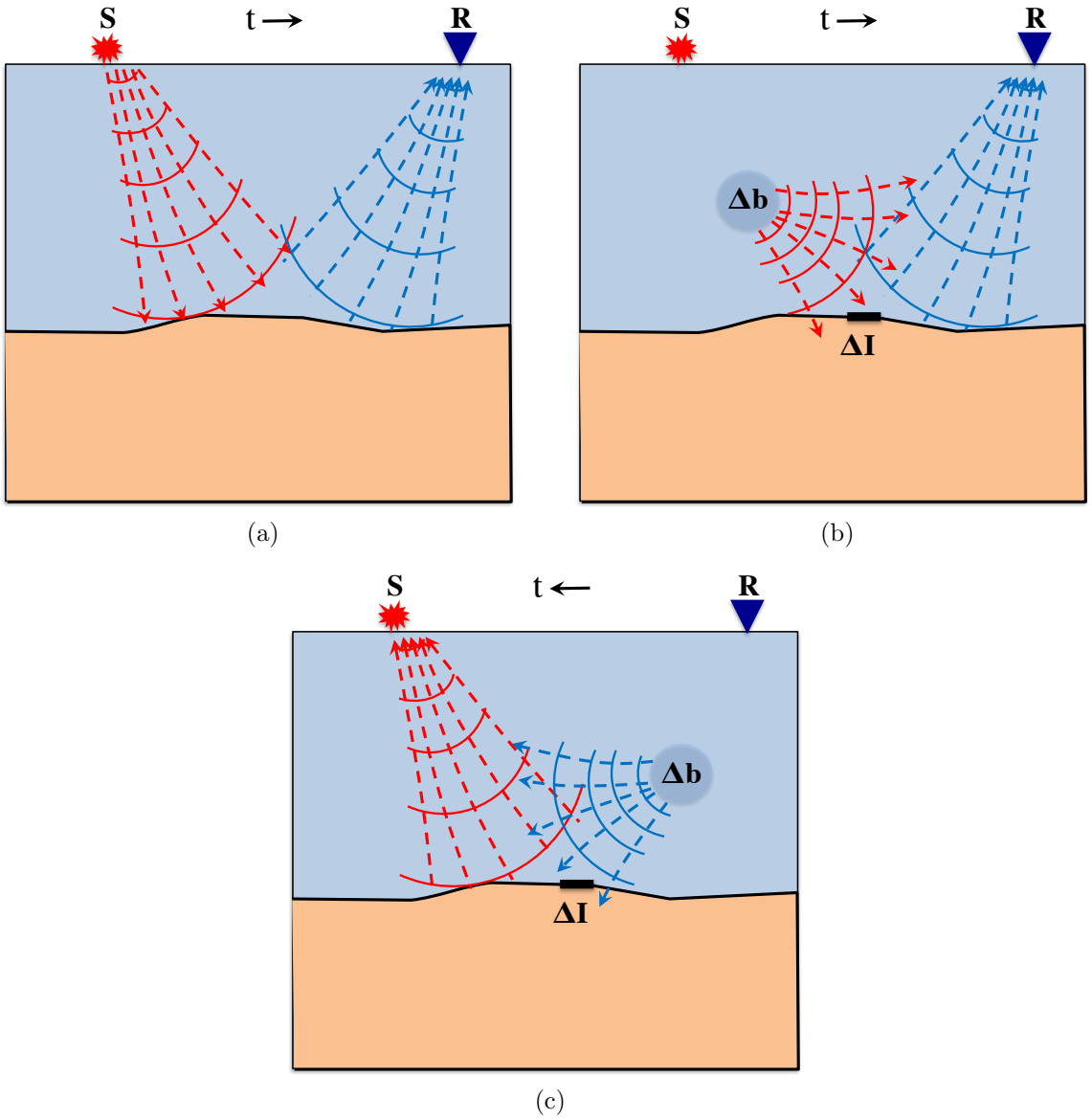

Figure 2: Stages of the forward WEMVA operator. a) Propagation of the background source and receiver wavefields. b) Propagation of scattered source wavefield and correlation with background receiver wavefield. c) Propagation of scattered receiver wavefield and correlation with the background source wavefield. The arrow in front of the symbol "t" represents time direction forwards (pointing right) or backward (pointing left). [**NR**]

Algorithm 1 WEMVA implementation saving both source and receiver wavefields

- Forward *propagate* the source wavefield and store; then, scatter upon perturbation and forward *propagate* the scattered source wavefield.
- Backward *propagate* the receiver wavefield and store; then, scatter upon perturbation and backward *propagate* the scattered receiver wavefield.
- Perform cross-correlations during propagations of the scattered wavefields.

This procedure demands four propagations (indicated in italics): two propagations for the background wavefields and two propagations for the scattered wavefields. Note that only the source and the receiver wavefields need be stored, not the scattered wavefields.

Implementation of the WEMVA operator storing one background wavefield

Let us assume that we can afford to store only one background wavefield, for instance, the source wavefield. In this case, we can proceed as indicated in Algorithm 2.

Algorithm 2 WEMVA implementation storing one wavefield at a time

- Forward *propagate* the source wavefield and store it.
- Backward *propagate* the receiver wavefield and scatter upon perturbation; then, backward *propagate* the scattered receiver wavefield on the fly.
- Cross-correlate the scattered receiver wavefield with the stored source wavefield, whereas propagating the former.
- Delete the source wavefield.
- Backward *propagate* the receiver wavefield and store it.
- Forward *propagate* the source wavefield and scatter upon perturbation; then, forward *propagate* the scattered source wavefield on the fly.
- Cross-correlate the scattered source wavefield with the stored receiver wavefield, whereas propagating the former.

Notice that now we need to perform two additional propagations compared to the previous case. This is the price that we pay when we avoid storing one background wavefield.

Implementation of the WEMVA operator with RBC

One step further to prevent the storage of the background wavefields is using RBC to ensure the reversibility of propagations. We can proceed as indicated in Algorithm 3.

Algorithm 3 WEMVA implementation storing none of the wavefields (using RBC)

- Forward *propagate* the source wavefield; then save the last two time frames.
- Backward *propagate* the receiver wavefield and scatter upon perturbation; then, backward *propagate* the scattered receiver wavefield "on the fly".
- At the same time, backward *re-propagate* the source wavefield and cross-correlate with the scattered receiver wavefield. Save the last two frames of the receiver wavefield.
- Forward *propagate* the source wavefield and scatter upon perturbation; then, forward *propagate* the scattered source wavefield "on the fly".
- At the same time, forward *re-propagate* the receiver wavefield and cross-correlate with the scattered source wavefield.

The cost of not saving wavefields at all is performing three further propagations with respect to the case of storing both wavefields. In total, we need to perform seven propagations. This scheme is the one that I employ in the computational codes that accompany this thesis.