Reverse time migration: Saving the boundaries

Robert G. Clapp

ABSTRACT

The need to save or regenerate the source or receiver wavefield is one of the computational challenges of Reverse Time Migration (RTM). The wavefield at each time step can be saved at the edge of the damping/boundary condition zone. The wave equation can be run in reverse, re-injecting these saved points to regenerate the wavefield. I show that this a better choice than checkpoint schemes as the domain grows larger and if the computation is performed on a streaming architecture.

INTRODUCTION

Reverse time migration (Baysal et al., 1983; Etgen, 1986) is quickly becoming the high-end imaging method of choice in complex geology. One of the computational challenges of RTM is that the source wavefield is propagated forward in time (0 to $t_{\max}$) while the receiver wavefield is propagated backwards in time ($t_{\max}$ to 0), yet the imaging step requires these two fields to be correlated at each time t. Storing one of the wavefields in memory is impractical for 3-D problems. The most obvious solution is to store the wavefield at each imaging step on disk. This requires significant disk storage on each node and can cause the problem to quickly become Input/Output (IO) bound. Symes (2007) proposed a checkpointing scheme where a smaller number of snapshots are stored to disk and intermediate wavefield are regenerated. Another approach, alluded to in Dussaud et al. (2008), is to save the wavefield at boundaries of the computational domain and to re-inject them.

In this paper, I demonstrate how to implement a boundary re-injection scheme. I discuss the computational tradeoffs of boundary re-injection vs. a checkpoint scheme and conclude that, as the computational domain increases in size, the boundary method proves superior.

REVERSE TIME MIGRATION REVIEW

Reverse time migration is an attractive imaging method because it does not suffer from the limitations of the Kirchhoff and downward continuation based imaging approaches. A single, or even multiple, travel-time can not accurately describe wave propagation in complex media, making Kirchhoff methods ineffective under salt. The

angle limitation, and difficulty/inability to handle multiple bounces limits the effectiveness of downward continuation based methods with complex structures.

Algorithm

The basic idea behind RTM is fairly simple: we are reversing the propagation experiment. Given data d recorded from $1...nt$ with sampling dt, we begin by re-injecting into our earth model that data recorded at nt. We then propagate this data back into the earth a length in time dt. We can imagine storing the current status of the receiver wavefield W_r into the last elements of 4-D volume in terms of x, y, z, t. We then re-inject at the surface data recorded at $nt - 1$. We follow this procedure until we have re-injected to the first time sample.

We also need to simulate the source portion of the experiment. We input at the source location of the earth model the first sample of the source pulse. We then propagate the wavefield dt, insert the next sample of the source pulse and continue until we have reached $dt * nt$ time. At each dt we store in a second wavefield array W_s the wavefield at each time sample. The zero offset migrated image I is formed at each iz, ix, iy location by

$$I(ix, iy, iz) = \sum_{it=0}^{nt} W_s(ix, iy, iz, it) W_r(ix, iy, iz, it). \tag{1}$$

.

Cost

The major disadvantage of RTM is that the cost is generally thought to be an order of magnitude more expensive than wave equation or Kirchoff alternative. Calculating the cost of the various imaging methods is a tricky proposition. In general people use two different metrics for comparing cost. The first is to simply count operations; the second counts the number of memory access requests. Which method is appropriate depends on whether you are saturating the memory bus or computation units. Kirchhoff migration is more likely to be limited by memory access, while downward continuation is usually limited by the computational units.

In today's world of multi-core, many-core, streaming architectures a third metric needs to be applied: how parallelizable is the method? This is in many ways a more difficult assessment because it strongly depends on the type of hardware and the type of parallelization. Many-core and streaming architectures are most effective with problems that are compute-bound and can have a high level of fine-grained parallelism. **?** showed how downward-continuation based methods can be limited by the data-dependency of the FFT.

The main cost of RTM is advancing the wavefield in time. This is usually done by solving the wave-equation using finite-differences. The basic idea is to start from

some version of the wave equation, for example the acoustic version where u is the wavefield,

$$\frac{\partial^2 u}{\partial t^2} = v^2(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}). \tag{2}$$

Each second derivative is approximated by doing a Taylor (or similar method) expansion. The difference approximation affects the accuracy of the propagation along with the required sampling of the wavefield in time and space to maintain stability and avoid dispersion. In general, a much higher order accuracy approximation is used in the space domain than the time domain, often from 6th to 14th order. The filter implied by this approximation can be quite large, from 19 to 43 points in size, and is the dominant computational cost. On the other hand, the structure of the computation is quite simple with virtually no data-dependency, and amenable to a very high level of fine-gain parallelism. As a result it is ideal for many-core and streaming architectures. The speed advantage of FFT and sparse matrix (Kirchoff) approaches diminishes significantly with these platforms.

As the speed of the wavefield propagation increases, a new bottleneck appears in a standard RTM implementation. First, the size of W_r and W_s are well beyond a conventional system's memory. In addition, note how W_r and W_s are filled in reverse order. The receiver wavefield stores $nt, nt-1, ...1$ while the source is filled $1, 2, ...nt-1, nt$. The obvious solution is to store one of the two fields (from now on I will choose the receiver wavefield) to disk. This requires a large but feasible amount of storage. The problem is that writing to disk is orders of magnitude slower than accessing memory. With an optimized implementation of the propagator, reading and writing the wavefield becomes the bottleneck.

The most common solution to this problem is to save a subset of the receiver wavefields. **?** describes an 'optimal' checkpointing scheme which minimizes the total number of imaging wavefields that need to be recomputed by storing a series of checkpoints either in main memory or on disk. A similar approach is linear checkpointing which stores every jt imaging steps where jt is a function of the amount of memory on the system.

SAVING THE BOUNDARY

Another approach is regenerating the wavefield, taking advantage of the reversibility of the wave equation. For example, imagine using a second order in time finite-difference scheme. Given the wavefield at the current time w_t and previous time $w_{t-\Delta t}$ we can find the wavefield at the next time $w_{t+\Delta t}$ through

$$w_{t+\Delta t} = w_{t-\Delta t} + w_t + Lw_t, \tag{3}$$

where L calculates the second derivative. We can reverse $w_{t-\Delta t}$ and w_t in equation 4 to find thew wavefield at $w_{t-2\Delta t}$,

$$w_{t-2\Delta t} = w_t + w_{t-\Delta t} + Lw_{t-\Delta t}. \tag{4}$$

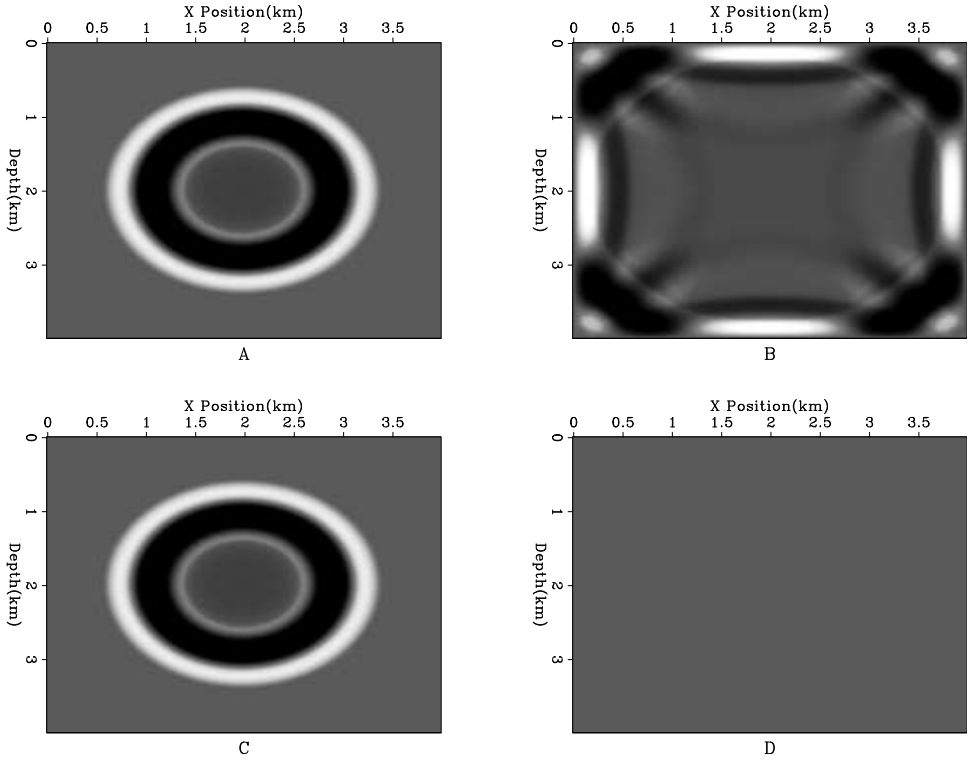

Figure 1: Panel 'A' shows a wavefield at time t, panel 'B' shows the wavefield after $2t$ at which stage the calculation is reversed. Panel 'C' shows the regenerated data, again at time t. Panel 'D' shows the difference between the regenerated wavefield and the original wavefield. [**ER**]

Figure 1 demonstrates this property. Panel 'A' shows a wavefield at time t, panel 'B' shows the wavefield after $2t$ at which stage the calculation is reversed. Panel 'C' shows the regenerated data, again at time t. Panel 'D' shows the difference between the regenerated wavefield and the original wavefield (with the clip at 1/10th panels 'A','B', and 'C'. Note that even energy that has hit the boundary has been handled correctly.

The problem comes when we attempt to kill energy entering the boundary. No longer is our time reversal scheme valid, as equation 4 does not fully describe what is being applied to the wavefield. Figure 2 demonstrates this concept. In this figure a damping boundary condition has been applied. Note how the wavefield and the reconstructed wavefield vary significantly.

Figure 3 demonstrates a way to solve this problem. Imagine our first time reversal step. The grey area represents cells where a boundary condition is being applied. Any locations where the filter implied by $\mathbf{L}$ touches a grey region will lead to incorrect reverse propagation. The area in black in Figure 3 show regions which uses cells where the boundary condition has been applied. If we save the wavefield from the forward propagation in the black region, we can substitute them in when doing the

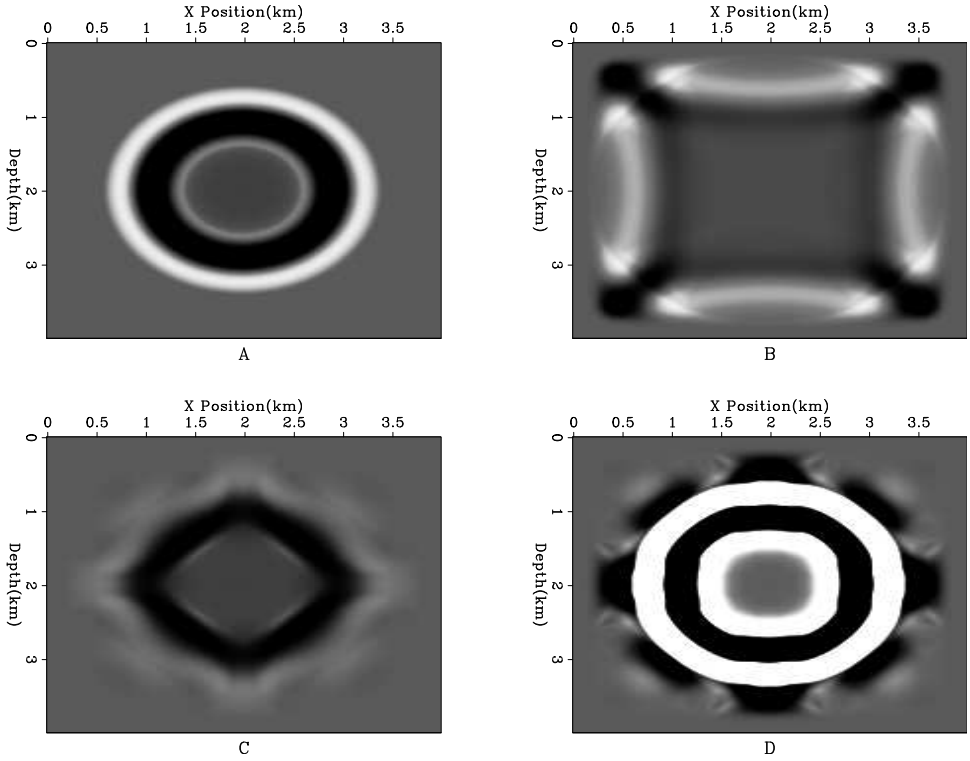

Figure 2: Panel 'A' shows a wavefield at time t, panel 'B' shows the wavefield after $2t$ at which stage the calculation is reversed. Panel 'C' shows the regenerated data, again at time t. Panel; 'D' shows the difference between the regenerated wavefield and the original wavefield. In this case a damping boundary condition has been applied around the edges of the domain. [**ER**]

reverse propagation. Figure 4 shows the result of this save and replace scheme. Note how we achieve a perfect result everywhere except the damping zone.

Note how the size of the black region is a single sample, while as previously noted, we tend to use a high order approximation in the space domain (which would make the black area larger). For this example I reduced the derivative approximation as I approached starting from 10th order solution and going to a second order at the boundary between the damped and undamped region. As a result only a single point along the boundary sees the damped region.

BOUNDARY VS. CHECK-POINTING

There are several metrics in comparing the 'cost' of using checkpointing vs. boundary saving approach. These are the amount of disk IO, IO throughput required, the amount of main memory needed for an optimal solution, and the ease of implementation. I will compare the linear checkpointing vs. boundary saving approach, but the optimal approach would behave similarly.

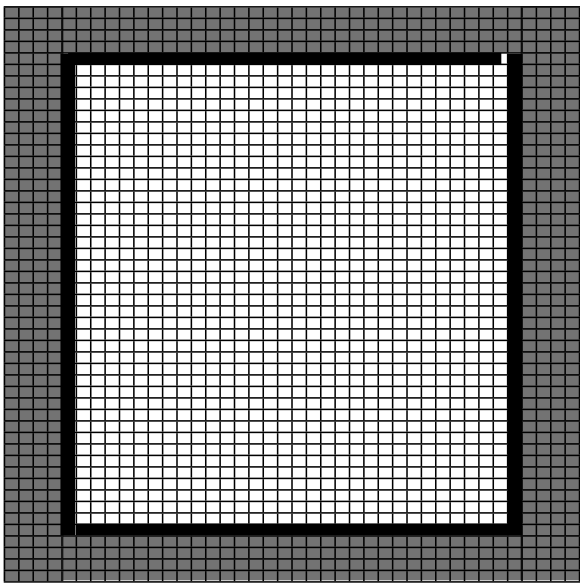

Figure 3: Computational grid for propagation. Grey area is cells effected by applying boundary conditions. Black area is cells that are saved and re-injected. [**ER**]

To setup the comparison let's assume that n is the length of the domain in x, y, and z. We have nt propagation steps, ji is the the number of propagation steps between imaging steps, and for the checkpointing scheme we will save every jc imaging time. Each approach does the same amount of propagation step $nt * 3$. The amount of disk required for the checkpointing m_c scheme is

$$m_c = \frac{2 * nt * n^3}{ji * jc}. \tag{5}$$

For saving the boundaries we need to save slices around the edge of the cube rather than the entire cube, but we need to save at every time step. The memory requirement m_b is then

$$m_b = 6nt * n^2. \tag{6}$$

Checkpointing requires less disk when we save less than every $\frac{n}{3*jt*jc}$ steps. Put another way, the larger the migration aperture the more advantageous saving the boundaries.

Checkpointing requires larger volumes, but less frequent reads; buffering read requests are a necessity with a checkpoint approach, but add to memory requirements. For true streaming hardware this can still be problematic because the entire buffer must be re-passed to the streaming engine. The boundary approach requires a much smaller volume to be passed continuously.

The memory requirement of the two systems is significantly different. The checkpointing scheme must redo the propagation in the same direction or suffer problems at the boundaries. This means that we must save jc copies of wavefield volume in memory, exchanging disk space for memory space. Again, this can be problematic if the imaging step is done with a hardware accelerator. It requires the volume to be read from disk and passed to the accelerator and each imaging step to be passed back

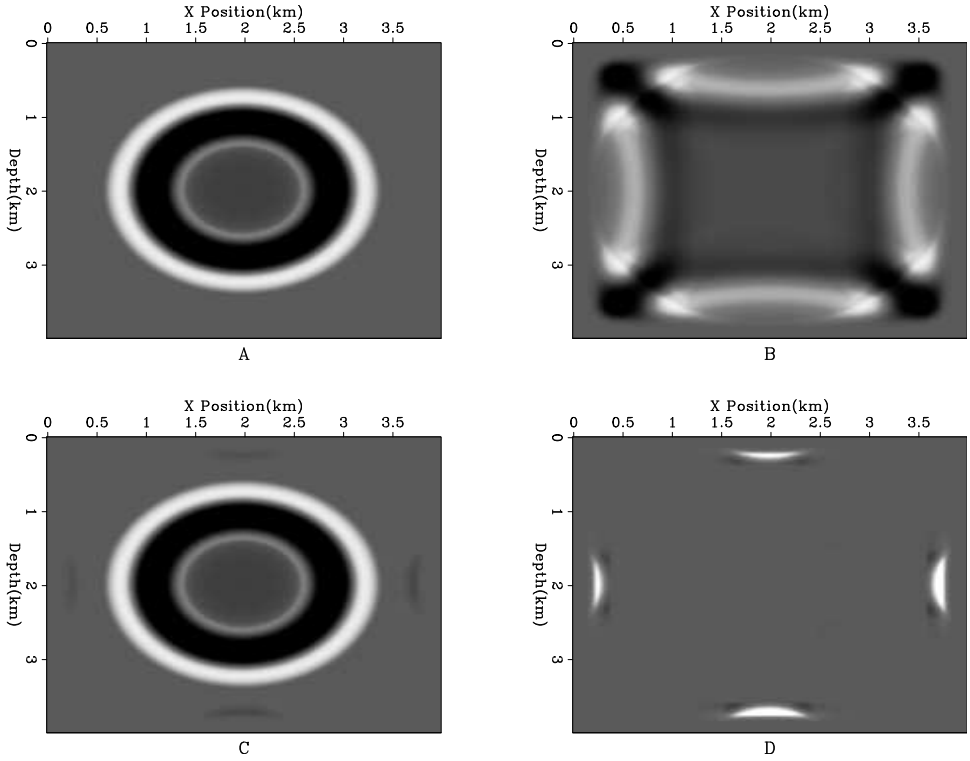

Figure 4: Panel 'A' shows a wavefield at time t, panel 'B' shows the wavefield after $2t$ at which stage the calculation is reversed. Panel 'C' shows the regenerated data, again at time t. Panel 'D' shows the difference between the regenerated wavefield and the original wavefield. In this case a damping boundary condition has been applied around the edges of the domain and the wavefield within this region has been re-injected. [**ER**]

to main memory, then sent back to that accelerator for producing the image. In the boundary saving approach the computations are done in the same directions, greatly reducing communication requirements.

Neither scheme is difficult to implement on a conventional CPU. Checkpointing benefits more from smart overlapping IO and compute but this doesn't add significant complexity. On accelerators the checkpointing scheme, with its significant additional data movement, is significantly more difficult to implement and optimize.

CONCLUSIONS

RTM lends itself well to fine grain parallelism and hardware acceleration technologies. Disk IO becomes the bottleneck with faster wavefield propagation. Checkpointing and saving boundary regions can reduce the IO cost while increasing the compute cost. For large problem saving the boundary is the more efficient mechanism.

REFERENCES

Baysal, E., D. D. Kosloff, and J. W. C. Sherwood, 1983, Reverse time migration: Geophysics, **48**, 1514–1524.

Dussaud, E., W. W. Symes, L. Lemaistre, P. Singer, B. Denel, and A. Cherrett, 2008, Computational strategies for reverse-time migration: 78th Annual Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts, SPMI 3.3.

Etgen, J., 1986, Prestack reverse time migration of shot profiles: SEP-Report, **50**, 151–170.

Symes, W. W., 2007, Reverse time migration with optimal checkpointing: Geophysics, **72**, SM213–SM221.