Stanford Exploration Project, Report 79, November 16, 1997, pages 83–91

Stanford Exploration Project, Report 79, November 16, 1997, pages 83–91

A parallel implementation of Kirchhoff DMO

Patrick Blondel[1]

keywords: *approximation, DMO, processing, three-dimensional, algorithm, elastic, modeling, linear*

ABSTRACT

In this paper I compare two parallel algorithms for three-dimensional (3-D) Kirchhoff dip moveout (DMO) in a constant velocity medium. In the case of 3-D land data, the highly irregular offset geometry pleads for an algorithm where data are processed in time slices. This mapping of the data into processor memory minimizes the cost of communication, which is reduced to nearest neighbor communication of time slices, and achieves load balance, keeping eighty percent of the processors busy throughout the process. The time aliasing of the operator is solved by a spatial convolution with dip-dependent triangles in a two-dimensional parallel implementation.

INTRODUCTION

Over the last years, DMO has become a standard step in seismic processing flows. The main reason for this is that it clearly improves the stack for little extra computational cost. Recently, some research showed that the improvement is even more distinct when depth-variable velocity is considered in the DMO process (Godfrey, 1992; Artley, 1992; Meinardus and Schleicher, 1993). However, these new methods are computationally expensive and a constant-velocity DMO is often enough to give an idea of the geological structures involved or improve the velocity analysis. Therefore, a parallel implementation of constant-velocity DMO is natural to reduce the processing time cost.

In a 3-D constant-velocity medium, the DMO process uses a line operator (Hale, 1991). It convolves the input data with elliptical impulse responses, and sums the convolved data into the final stacked volume. The single dimension of the operator as well as its dip limitation are two reasons to consider 3-D constant-velocity DMO as a fast process whatever the geometry of the data acquisition. Unfortunately, the multi-azimuthal distribution of 3-D land data is a major obstacle for parallel implementation (Biondi and Moorhead, 1992).

[1]**email:** not available

This paper details two possible parallel algorithms for 3-D constant-velocity DMO. The first algorithm involves a spiral data movement that does not use the full computational capabilities of the connection machine (CM5). The second algorithm reduces the communication cost by processing time slices but requires additional care to avoid temporal aliasing of the operator.

DMO BY SPIRAL DATA MOVEMENT

The parallel implementation of DMO in a two-dimensional space is straightforward. Biondi (1991) describes a simple algorithm where the traces are laid out local to each processor. The DMO process then consists of a trace stretch and a nearest neighbor shift along the midpoint axis.

In three dimensions, the data shift is not restricted to a single axis but requires a movement of the input traces across a two-dimensional space. Biondi (1992) uses the same layout to apply DMO to 3-D marine data. The algorithm takes advantage of the regular data movement in the in-line direction because of the uniform sampling of the shot locations. However, 3-D land surveys record shot profiles of varying geometry (an example is shown in Figure **??**), and thus, the shot axis can not be processed in parallel.

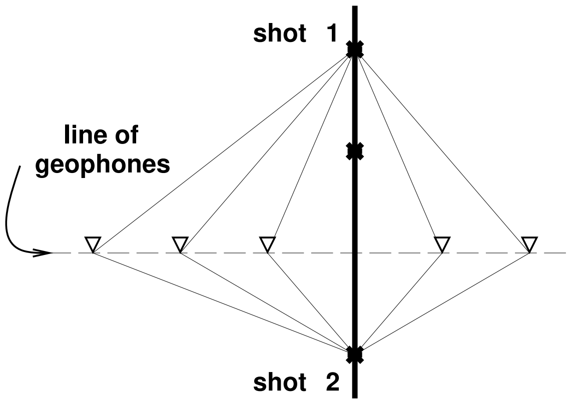


Figure 1: This simple 3-D land acquisition geometry exhibits the non-recurrence of the shot profiles. As opposed to marine surveys, the geophone line does not move along with the shot location. patrick1-land3d [NR]

Memory layout

Since the irregularity of 3-D land data appears in both azimuth and offset distributions, there is no prreferential direction for data movement. Hence, Biondi had the idea of a spiral data movement alternating the shifts along the x- and y- axes. The algorithm can be outlined as follows:

1. The data are sorted by offset length into classes.
2. For each bin of the output space, the trace of the current class whose midpoint falls into the bin is loaded in-processor.

3. The DMO stretch is performed in parallel over output space.

4. The traces are moved to the neighboring processor according to a spiral pattern illustrated in Figure **??**.

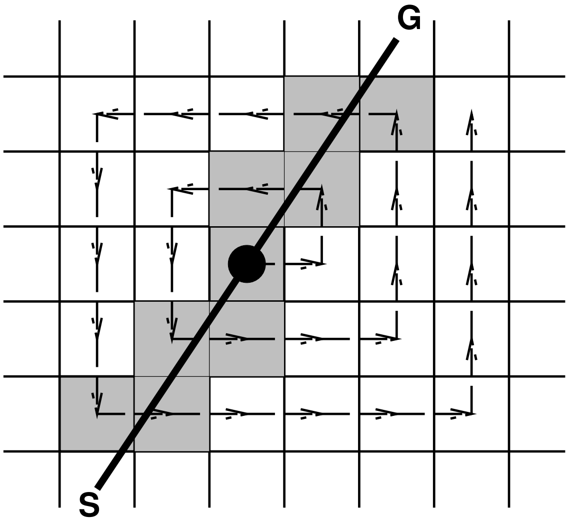


Figure 2: Spiral pattern for data movement over output space. The shaded squares represent the processors that actually perform the stretch and the stack to the output. The white squares represent the processors that stay idle during the movement of the trace. patrick1-spiral [NR]

Load balance

By sorting the data into classes of offset, we can restrict the size of the spiral to the maximum offset of each class. The spiral data movement allows any azimuth distribution of the input traces. However, because a trace contributes to the stack only along its original source-geophone line, some processors are left idle during the spiral trip of the trace. Figure **??** shows the working processors in grey and the idle processors in white. In this example, the computer load, defined as the ratio of the number of working processors to the total number of processors, is thirty percent. For larger offsets, the load balance is expected to be worse because the number of working processors increases in proportion to the offset length whereas the total number of processors increases in proportion to the square of the offset. The unoptimized use of processing power makes the algorithm inefficient for 3-D land data.

DMO BY TIME SLICE

In order to spread the data along an elliptical path in (t, x, y) space, it is possible to either move spatially from bin to bin and stretch the traces or move the time slices upward and stretch the data in the offset direction.

Memory layout

The memory layout for processing time slices is illustrated in Figure **??**. The time slices are local to each processor and the time axis is the parallel dimension. Thus, the number of processors needed is the number of time samples, and the memory of each processor must be large enough to load and process one time slice. When the amount of data exceeds the memory available, the process can run on lumps of data. The lumps may be pieces of data cut in the (x, y) space, preserving the parallel axis and relieving the processors memory, or they may be cut in time, shortening the parallel axis but requiring a smaller area of overlap between successive lumps.

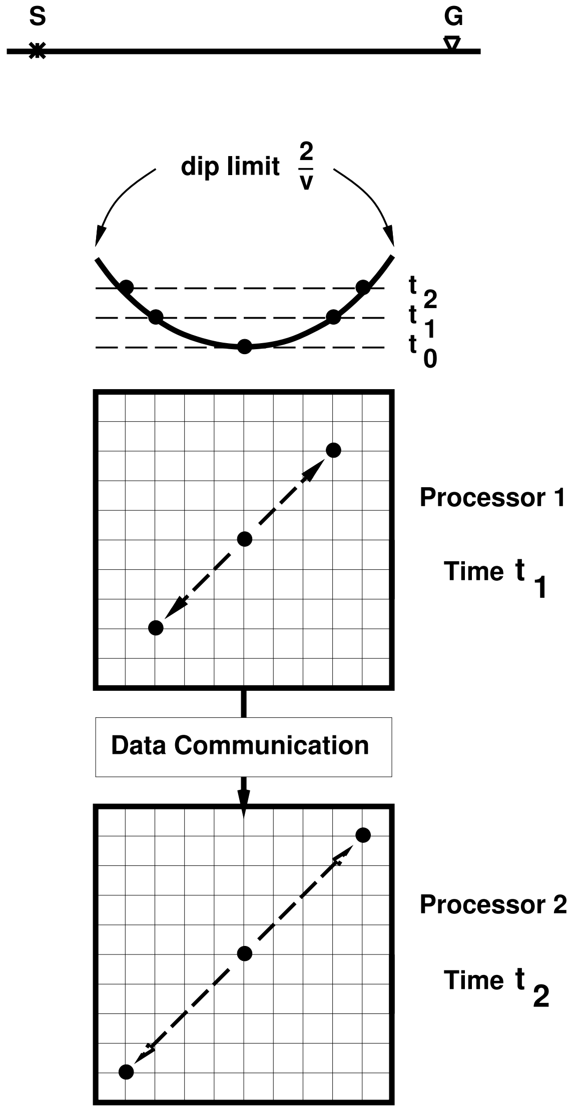


Figure 3: The top-most drawing represents the elliptic dip-limited DMO operator. I assume that the offset line bisects the x- and y-axes on the Earth's surface. Below, the two grids represent the data layout inside the processors. Each processor contains a time slice of data. Processor 1 contains the time slice at $t = t_0$ and performs the data communication across the (x, y) space that corresponds to a vertical shift from t_0 to t_1. During this time, processor 2 performs the same kind of operation for a vertical shift from t_1 to t_2. Then, processor 1 sums its output in the output volume and communicates its input to processor 2. The action is then repeated, moving the data across the (x, y) space for a vertical shift from t_0 to t_2. patrick1-tsproc [NR]

Unlike the previous algorithm, data communication is performed in a single direction, up the time axis, and does not depend on the offset and azimuth distribution. The long ranging and chaotic communications in the (x, y) space take place in-processor. Thus, the processing of time slices results in a more efficient inter-processor communication than the trace processing.

Load balance

Because the DMO elliptic operator is dip-limited, it does not extend all the way up to the Earth's surface ($t = 0$). The time spread of the impulse response is given by:

$$\Delta t = t_n \left(1 - \frac{1}{\sqrt{1 + \left(\frac{t_m}{t_n}\right)^2}} \right), \tag{1}$$

where t_n is the time location of the impulse in the input space, h is the half offset, v is the velocity of the medium, and $t_m = 2h/v$ is the horizontal two-way traveltime between shot and midpoint. Figure **??** is a plot of the time spread as a function of NMO time.

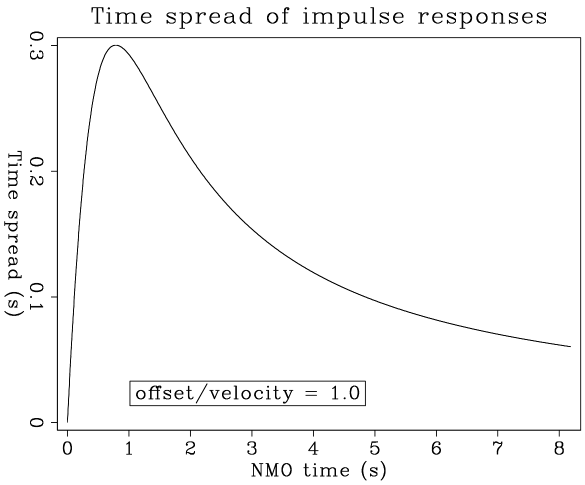


Figure 4: Time spread of the impulse responses of DMO as a function of the impulse location, t_n. The maximum time spread occurs for the input time $t_n = t_m/\sqrt{G}$ where G is the golden number. Click on the following button to see the curve for different times t_m. patrick1-magic [ER]

An interesting feature of these curves is that the time spread of the impulse responses never exceeds μt_m where μ is

$$\mu = \sqrt{G}\left(\frac{G-1}{G+1}\right) \simeq .300283106, \qquad \text{with } \mathrm{G} = \frac{1+\sqrt{5}}{2}. \tag{2}$$

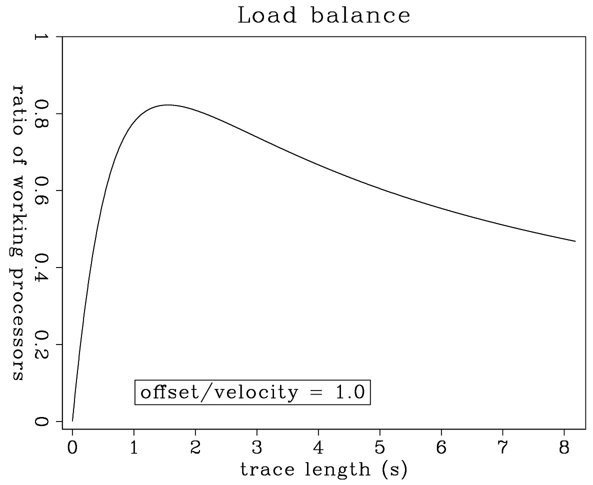
Because t_m is limited by the maximum offset, the time spread is much shorter than the full trace length.

During the process, the time slices are shifted upward until they reach the maximum time spread. Of course, only the time slice corresponding to the maximum time spread will have to be processed all the way. Other time slices, like for example $t_n = 5$ seconds (Figure **??**), will be processed for the first .1 second and then pass through idle processors. Obviously, the later time slices require less processing than the earlier ones, and thus represent a waste of processing capacity. The following formula gives the load balance as a function of trace length:

$$\mathrm{Load}(t_{\max}) = \frac{1}{t_{\max}} \int_0^{t_{\max}} \frac{\Delta t(t)}{\Delta t_{\max}} dt. \tag{3}$$

This formula is derived from the two following observations. The time for which the processors are active is proportional to the area under the curve representing $\Delta t(t)$ in Figure **??**, $\int \Delta t(t)dt$. The total computation time is proportional to both the trace length and the maximum time shift, $t_{\max}\Delta t_{\max}$. The computer load is then the ratio of the effective working time to the total execution time, as expressed in equation (3). As an example, for a trace length of 4 seconds and a midpoint time $t_m = 1$ second, the load balance reaches seventy percent (Figure **??**).

Figure 5: Load balance as a function of the trace length. The optimal load balance of eighty percent corresponds to a trace length which is a function of t_m $(= 2h/v)$. The bigger t_m is, the later the load balance is optimal. Click on the following button to see a movie of the load balance for different values of t_m. $\boxed{\text{patrick1-integ}}$ [ER]



This algorithm allows a more efficient distribution of work between processors than the spiral trace processing described earlier. I implemented this algorithm for a two-dimensional model (Figure **??** is an output of the program) but the run time is similar to a serial implementation of DMO (in 2-D, trace processing is more straightforward than time slice spreading). However, the real advantage of the method will appear in processing 3-D land data.

ALIASING OF THE OPERATOR

Applying DMO in time slices assures that the operator will not introduce spatial aliasing in the data. However, the spatial spreading from a time slice to the next must never exceed the bin spacing in order to avoid temporal aliasing (Hale, 1991). The previous condition is not verified for the gentle dips of the operator.

A technique described by Claerbout (1992) prevents operator aliasing. The method consists of the convolution of the operator with triangles whose width depends on the dip of the operator. The convolution is applied horizontally in the time slices. Figure **??** shows the DMO response of several impulses corrected with anti-aliasing triangles.

In three dimensions, the triangles cannot be obtained by Clearbout's double integration technique and thus, they are pre-computed and convolved explicitly.

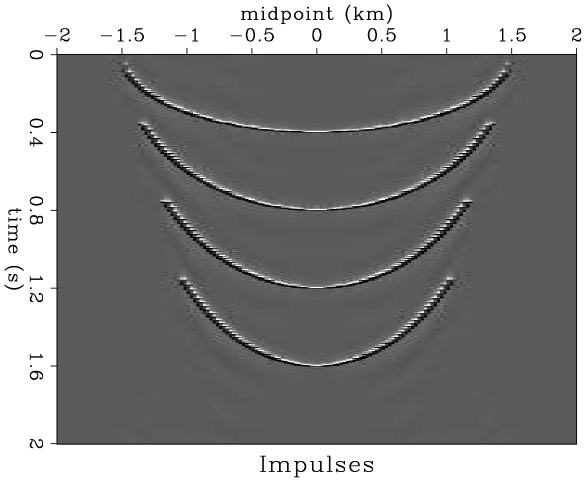


Figure 6: Impulse responses of DMO applied to time slices. The anti aliasing triangles are wide at gentle dips and shrink towards the steepest dips. patrick1-res [ER]

CONCLUSION

Because constant-velocity DMO in a 3-D Earth model operates along curves, care must be taken in parallel implementation in order to avoid the waste of computational capacity. Trace processing algorithms require a great deal of inter-processor communications and leave many processors idle during the process. On the other hand, the processing of time slices is well adapted to the irregular geometry of 3-D land data, reaching a load balance of eighty percent. Although the anti-aliasing convolution is more quickly performed along traces, the efficient utilization of computational capacity compensates for a slower convolution in the time slices.

The efficiency of the time slice processing has not been verified in three dimensions yet; however, the results of this study are encouraging.

REFERENCES

Artley, C. T., 1992, Dip moveout processing for depth-variable velocity: Master's thesis, Colorado School of Mines.

Biondi, B., and Moorhead, W. D., 1992, Data parallel algorithm for Kirchhoff 3-D DMO: 62nd Ann. Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts, 330–333.

Biondi, B., 1991, Wave equation algorithms on massively parallel computers: SEP–**70**, 59–72.

Claerbout, J. F., 1992, Anti aliasing: SEP–**73**, 371–390.

Godfrey, R. J., 1992, Dmo and $v(z)$: 62nd Ann. Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts, 952–954.

Hale, D., 1991, Course notes: Dip moveout processing: Soc. Expl. Geophys., Tulsa.

Meinardus, H. A., and Schleicher, K. L., 1993, 3-d time-variant dip moveout by the f-k method: Geophysics, **58**, 1030–1041.