Stanford Exploration Project, Report 73, November 18, 1997, pages 101–121

Reservoir velocity analysis on the Connection Machine

David E. Lumley[1]

keywords: *velocity, stacking, three-dimensional, algorithm, prestack, gas*

ABSTRACT

I am developing techniques to perform a detailed reservoir characterization given 3-D prestack surface seismic data. In this paper, I develop and implement two velocity analysis algorithms on an 8k processor Connection Machine CM-2. The first algorithm performs a weighted semblance stacking velocity analysis, and is general for 3-D bin geometries. It is heavily I/O bound, and the inner computational core currently runs at about 120 Mflop/s. The second algorithm performs a full 3-D prestack time migration velocity semblance analysis. It is mostly cpu bound and currently runs at about 420 Mflop/s. These results are expected to scale fairly linearly on a full 64k processor CM-2 to 1 Gflop/s for stacking velocity analysis and over 3 Gflop/s for prestack time migration velocity analysis. I tested the algorithm on a marine data set recorded over a producing gas reservoir in the Gulf of Mexico. This example shows a variation of at least 100 m/s in migration velocity, or about 500 m/s in interval velocity, in about 700 m distance along the gas reservoir unit. The estimated velocity variation correlates with AVO bright and dim spots, suggesting a technique for mapping 3-D spatial reservoir gas saturation and hydrocarbon pore volume.

INTRODUCTION

I am interested in 3-D prestack seismic reservoir characterization (see Lumley, 1991). I am actively seeking a 3-D prestack surface seismic data set, shot over an existing reservoir in a development and/or production stage. An outline of the general project goals consists of the following steps:

- Perform a high spatial resolution 3-D prestack time migration velocity analysis. This cube of velocity information $v(x, y, \tau)$ can be mapped out at every CMP bin location, inverted and smoothed to interval velocity $v(x, y, z)$. Changes in velocity in the reservoir zone may correlate to spatial variation in reservoir

[1]**email:** not available

porosity ϕ, physical states such as temperature or pressure due to steam injection or fire-flood, or pore content such as gas or fluid.

- Perform a full 3-D prestack time migration. This step depends intimately on a careful migration velocity analysis as described above. Hopefully, the increased resolution in migration velocity and prestack migration focusing may better delineate the reservoir target, and better resolve any distributed changes in spatial reservoir properties. In particular, a high resolution 3-D image may be important for resolving lithologic discontinuities (pinch-outs and unconformities) and faults, which impact bulk reservoir permeability κ, and may directly indicate presence of in-place hydrocarbons, or EOR injectants such as steam, CO_2, or fire-flood.

- Perform a full 3-D prestack migration-inversion (true amplitude migration + AVO inversion). Using the velocity and structural results above, make 3-D volume maps of relative contrasts in P and S impedance throughout the reservoir. These maps may better resolve zones of anomalous porosity or permeability, and may correlate with other interesting rock/reservoir properties in the form of "attribute indicators" as a combination of the P and S maps.

- If enough well log data are available, perform a geostatistical uncertainty analysis of the results above, using spatial correlations of in situ reservoir property measurements conditioned to the baseline seismic survey, to predict reservoir flow paths and probability of spatial trends in measured reservoir properties.

Several items in each of these phases will likely involve some interaction with the Stanford Center for Reservoir Forecasting (SCRF), a consortium of geostatisticians and reservoir engineers, and the Stanford Rock and Borehole Project (SRB) rock physics consortium.

The critical first phase of this project, upon which all subsequent phases depend, requires a high resolution 3-D prestack migration velocity analysis. By "high resolution" I mean increased velocity resolution in comparison to conventional surface seismic velocity analyses, in terms of increased 3-D spatial and temporal sampling.

To understand the intensity of the computational requirements for such a high resolution reservoir velocity analysis, consider a small 3-D reservoir survey with a surface migration image area of about 4 km^2. Given a CMP spacing of 25 m both in- and cross-line (after interpolation), this represents about 6400 full-fold CMPs. If the fold is 48, the entire 3-D prestack data set represents 300,000 traces. Given a sampling interval of 2 ms and a 4 s record length, the data set requires 2.5 Gbytes of storage. If we migrate 48 velocities at each of the 6400 CMP bin locations to produce a migration velocity analysis at every 25 m spatial interval, and require 50 flops per input trace per output pixel, the total flop count is about 10^{16}, or 10,000 Teraflops. Implementing the required algorithm on a supercomputer at 1 Gflop/s would require about 4 cpu months to complete the job! Clearly, we need to decrease this cpu time by at least an order of magnitude. Decreasing the run time could be accomplished by: (a) a sparser spatial resolution of the velocity analysis, (b) a sparser temporal velocity resolution, (c) decreasing the number of migration velocities using a priori knowledge of stacking velocities, (d) decreasing the number of input seismic traces, and (e) using a faster computer and/or algorithm. For example, halving the number of migration velocities to 24, decimating the input traces to 8 ms, and performing the velocity analysis every 100 m reduces the job described above to about 24 cpu hours, which is much more manageable.

Due to the large computational effort required by migration velocity analysis, I decided to implement the process on our Connection Machine. The first section of this paper discusses the stacking velocity analysis I developed on the CM. The next section describes the CM prestack time migration velocity analysis. Both methods are based on the Kirchhoff approach. After introducing each method and its implementation on the CM, I show an application of each to a 2-D marine data set exhibiting a bright spot from the Gulf of Mexico, and discuss pertinent run times and I/O issues.

STACKING VELOCITY ANALYSIS

This section concerns the implementation of a stacking velocity semblance analysis on the CM. Velocity semblance panels are computed for each desired CMP bin location. A divergence correction, offset weighting, obliquity weighting, vertical smoothing and time slice normalization are incorporated.

The algorithm accepts seismic trace data organized into CMP gathers/bins. These are stored on the CM in a 3-D array: `data(nt,nh,nm)`, where `nt` is the number of time samples per trace, `nh` is the number of offsets per CMP bin, and `nm` is the number of CMPs to be processed. The `data` array is mapped out on the CM as parallel in the offset and midpoint dimensions, and serial in the temporal direction: `data(:serial, 1000:news,:news)`. The offset dimension is weighted by 1000, which instructs the CMF compiler to organize the offset traces in each CMP gather closely together in adjacent processors. This weight improves inter-processor communication during the `CSHIFT` operations, as discussed later.

The output is organized into velocity semblance panels per CMP location. These are stored on the CM in a 3-D array: `vpanels(ntau,nv,nm)`, where `ntau` is the number of output time samples per velocity trace (usually about `nt`/10), and `nv` is the number of stacking velocities per panel. The `vpanels` array is mapped out on the CM as parallel in the velocity and midpoint dimensions, and serial in the temporal direction: `vpanels(:serial,1000:news,:news)`. The velocity dimension is weighted by 1000 in order to organize the velocity traces in each semblance panel closely together in adjacent processors.

To optimize CM performance, the products `nh`·`nm` and `nv`·`nm` should be equal, and a multiple of 512. Their equality results in a 1:1 overlap of the input and output arrays `data` and `vpanels`, which is optimal for CM processor communications, and the factor of 512 ensures compiler parallelization optimality. For the marine data example in this paper, which consisted of a series of 48-fold CMP gathers, I chose `nh` = `nv` = 48, and `nm` = 32. For alternate fold data (e.g., 60-fold), I would choose `nh` to be the next higher power of two and pad each gather with zero traces. It is also very efficient to choose the number of stacking velocities `nv` to be equal to the fold `nh`, to keep the input and output arrays aligned. Deviation from these rules of thumb can result in a significant loss of performance. The number of gathers `nm` is actually determined by the amount of available memory. For a small memory machine like our present CM-2 configuration (8k processors, 64 Mb memory), I could process the entire data set in single blocks of 128 48-fold CMP gathers (2,000 samples per trace), by sequentially loading data blocks and iterating the velocity analysis processing sequence.

For each data block, the stacking velocity semblance panels are computed as follows. Each offset trace of every gather is stretched with an offset-dependent stacking velocity, and stacked into its appropriate semblance panel. This operation is done in parallel across all offsets and gathers, and serially in time.

Then, within each semblance panel, the velocity traces are shifted in a circular manner with the `CSHIFT` intrinsic, such that the lowest stacking velocity trace v_1 moves to the v_2 position in the same CMP bin, and the highest velocity trace v_n moves to the lowest velocity position v_1 in the *next* CMP bin. This shift is relative to the input traces. Before the shift, seismic trace s_2 resides in the same processor as velocity trace v_2, and so on. After the shift, seismic trace s_2 now shares a processor with velocity trace v_1, and so on. Now each offset trace in each CMP gather is stretched with its new velocity, and stacked into the semblance panel.

This process is repeated until all traces are back in their initial configuration, which requires `nh` circular shifts. Note that input data could be shifted, rather than the semblance panels, but it is quicker to shift the smaller of the two arrays to minimize interprocessor communication. Finally, an array of squared velocity stack energy is maintained to compute semblance, and offset weights. The incorporation of divergence corrections, vertical smoothing and time-slice normalization complicates the simplistic description given above.

MIGRATION VELOCITY ANALYSIS

This section concerns the implementation of a 3-D prestack time migration velocity semblance analysis on the CM. Velocity semblance panels are computed for each desired surface location. A migration divergence correction, obliquity weighting, vertical smoothing and time slice normalization are incorporated.

The algorithm accepts seismic trace data organized in any arbitrary manner. These are stored on the CM in a 2-D array: `data(nt,ntrc)`, where `nt` is the number of time samples per trace, and `ntrc` is the number of traces from a 2-D or 3-D survey to be loaded into a data block for processing. The `data` array is mapped out on the CM as parallel in the surface coordinate dimension, and serial in the temporal direction: `data(:serial,:news)`.

The output is organized into velocity semblance panels per surface location. These are stored on the CM in a 2-D array: `vpanels(ntau,nvtrc)`, where `ntau` is the number of output time samples per velocity trace (usually about `nt`/10). The dimension `nvtrc` is the total number of output velocity traces, which can be expressed as `nvtrc` = `nv`·`nx`·`ny`, where `nv` is the number of migration velocities per semblance panel, and `nx`, `ny` are the number of desired output velocity panels at 3-D in-line and cross-line surface locations respectively. The `vpanels` array is mapped out on the CM as parallel in the surface coordinate dimension, and serial in the temporal direction: `vpanels(:serial,:news)`.

To optimize CM performance, the products `ntrc` and `nvtrc` should be equal, and a multiple of 512. As in the stacking velocity case, their equality results in a 1:1 overlap of the input and output arrays `data` and `vpanels`, which is optimal for CM processor communications, and the factor of 512 ensures compiler parallelization optimality. For the marine data example in this paper, which consisted of a series of 48-fold CMP gathers, I chose `ntrc` = 48·32 = 1,536 input traces. In practical terms, the constraint on `ntrc` is determined by the output dimension length `nvtrc`. For the marine data, I chose `nv` = 48, `nx` = 32 (32 48-fold CMP gathers), and `ny` = 1 (2-D seismic data). In general, I would suggest making `nvtrc` as large as possible while still a multiple of 512. This involves deciding how many surface positions `nx` and `ny` at which to do a velocity analysis, and how many migration velocities `nv` to try at each surface position. Then I would set the number of input traces per data block equal to the total number of velocity traces: `ntrc` = `nvtrc`. Since the input seismic trace length is usually about 10 times longer than the output velocity trace length, some trade-off in total memory available has to be balanced in choosing these parameters.

Deviation from these rules of thumb can result in a significant loss of performance. For a small memory machine like our present CM-2 configuration (8k processors, 64 Mb memory), I could process the entire data set in single blocks of 6,144 arbitrarily gathered traces, at 2,000 samples per trace, by sequentially loading data blocks and iterating the velocity analysis processing sequence. This would yield an output velocity analysis at each of 128 surface positions, each semblance panel containing 48 migration velocities and 256 time samples.

The CM migration velocity analysis algorithm is based on a 3-D Kirchhoff prestack time migration algorithm previously developed for the CM (Lumley and Biondi, 1991). For each data block, the migration velocity semblance panels are computed as follows. Each input trace s_i is migrated into a single surface location x_i at a single migration velocity v_i. This step is accomplished in parallel over the trace dimension for a fixed time slice, and serially over all time slices. Once this step is complete, the output array `vpanels` and the velocity and output coordinate information are `CSHIFT`ed once over the parallel dimension, a shift that is relative to the input traces. After the shift, a seismic input trace s_1 that was aligned in-processor with velocity trace v_1 at output surface location x_1, will now be aligned in-processor with the next velocity trace v_2 at the same output location x_1. Similarly, a seismic input trace s_n that was aligned with velocity v_n at output surface location x_1, will now be aligned with the first velocity v_1 at the *next* output location x_2.

This process is repeated until all traces are back in their initial configuration, which requires `nh` circular shifts. Again, the input data could be shifted, rather than the semblance panels, but it is quicker to shift the smaller of the two arrays to minimize interprocessor communication. Finally, an array of squared velocity migration stack energy is maintained to compute semblance, and offset weights. Divergence corrections, vertical smoothing and time-slice normalization are incorporated which complicate the simplistic description given above.

MARINE DATA EXAMPLE

I now demonstrate the stacking velocity and migration velocity algorithms on some CMP gathers recorded in the Gulf of Mexico. In addition, I discuss cpu times and I/O behavior of the two algorithms with respect to this marine data example.

Figure **??** is a near offset section displaying a bright spot in the Gulf of Mexico. Figures **??** and **??** are the mid and far offset sections respectively. The offset distances are 241 m, 1391 m, and 2341 m. These three constant offset sections are representative of hydrocarbon reservoirs which exhibit spatially variable reservoir rock properties and/or pore constituents. The lateral variation in relative P impedance in the near offset section could be indicative of spatially variable gas content, and in turn, variable porosity, for example. Of course, some spatial amplitude variation is almost certainly due to near surface "overburden" focusing/defocusing effects, and would have to be compensated in any amplitude inversion process. The three sections also indicate a general AVO amplitude increase with offset, which is a common signature of HC gas. A migration-inversion of these data could potentially better resolve the spatial distribution of the reservoir properties which cause lateral variations in seismic amplitude (Lumley and Beydoun, 1991, and Parsons, 1986).

Figure **??** shows four CMP gathers extracted from midpoints in the 4-5 km distance range of the common offset sections. The strong AVO effect is readily visible in the gathers. Ideally, a high spatial resolution velocity analysis at a fine CMP interval spacing across the section might detect significant lateral variations in velocity. Given some in situ well log/core measurements, it may be possible to correlate such velocity variation to spatial variations of rock properties such as pore gas/fluid content, porosity or clay content (Han, 1986).

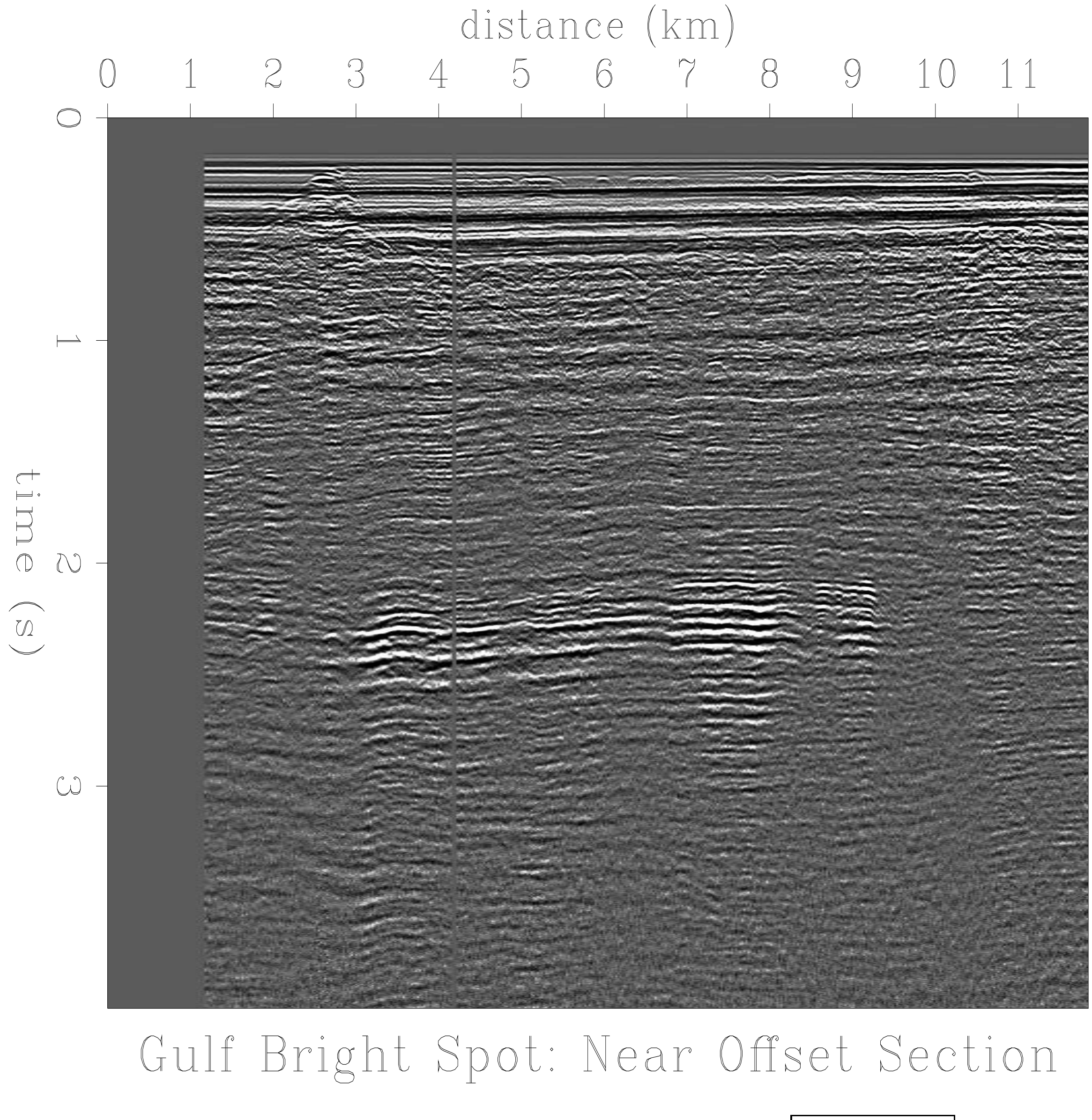

Figure 1: Near offset section. Constant offset is 241 m. david2-near [NR]

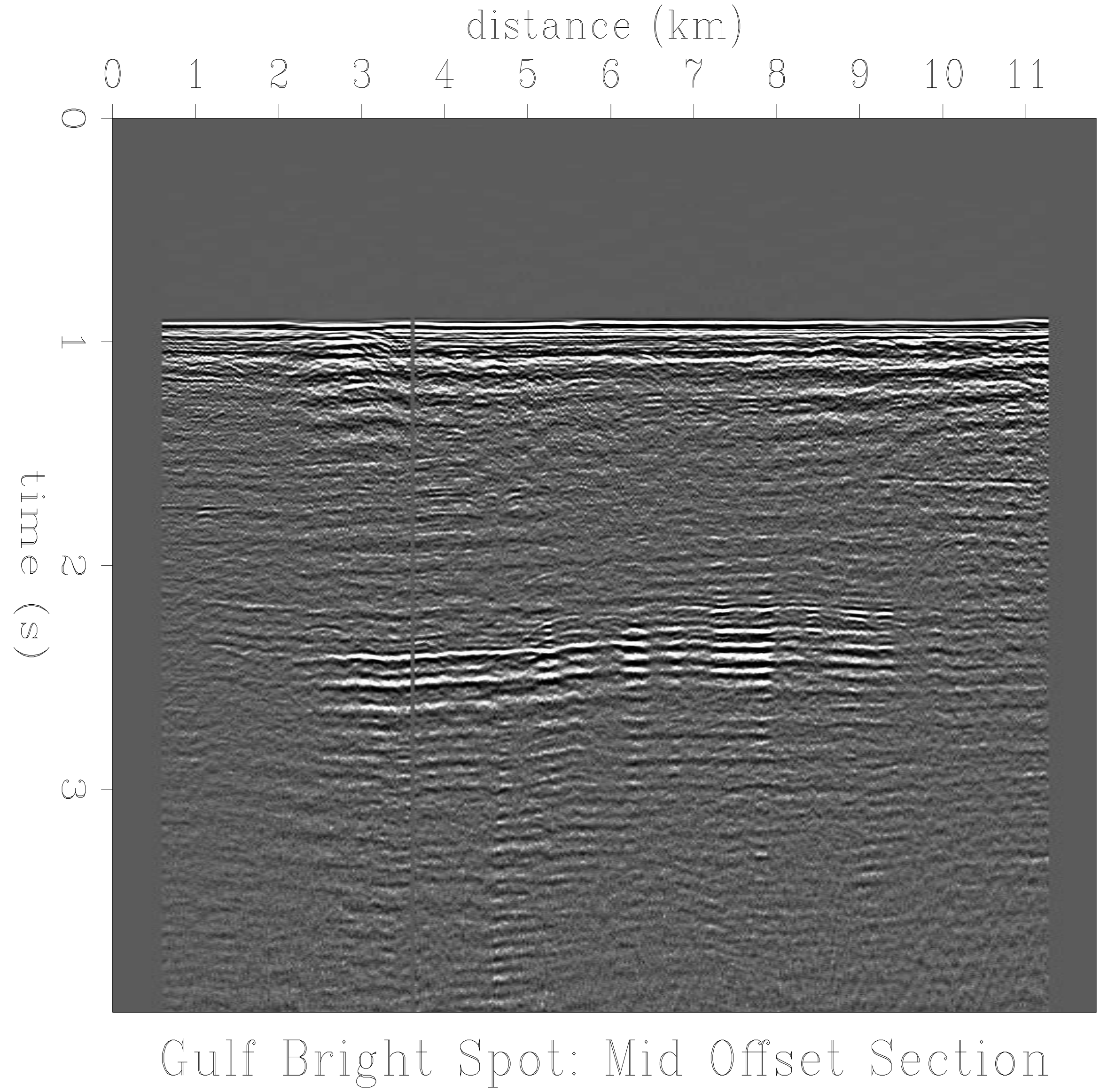

Figure 2: Mid offset section. Constant offset is 1391 m. $\boxed{\text{david2-mid}}$ [NR]

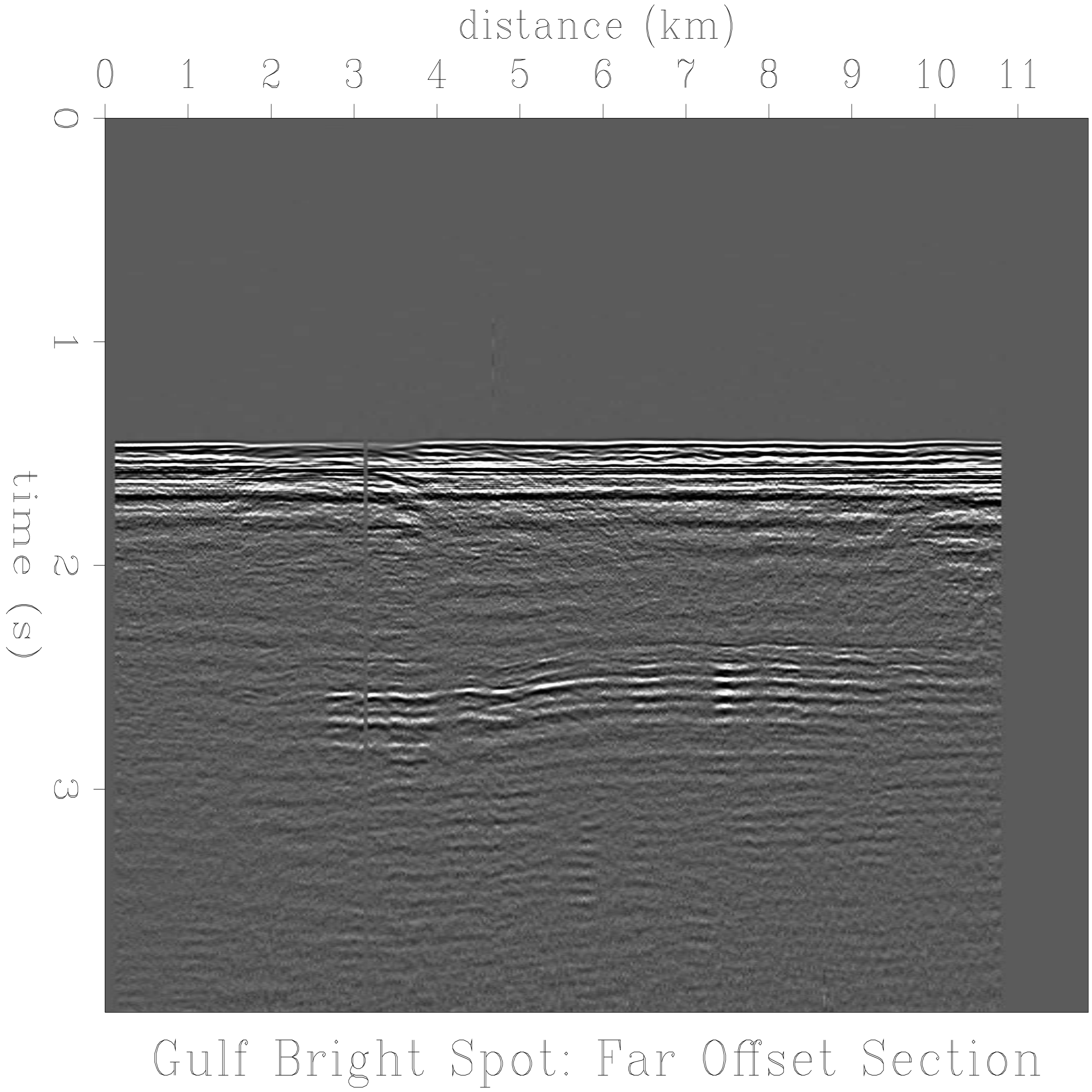

Figure 3: Far offset section. Constant offset is 2341 m. $\boxed{\text{david2-far}}$ [NR]

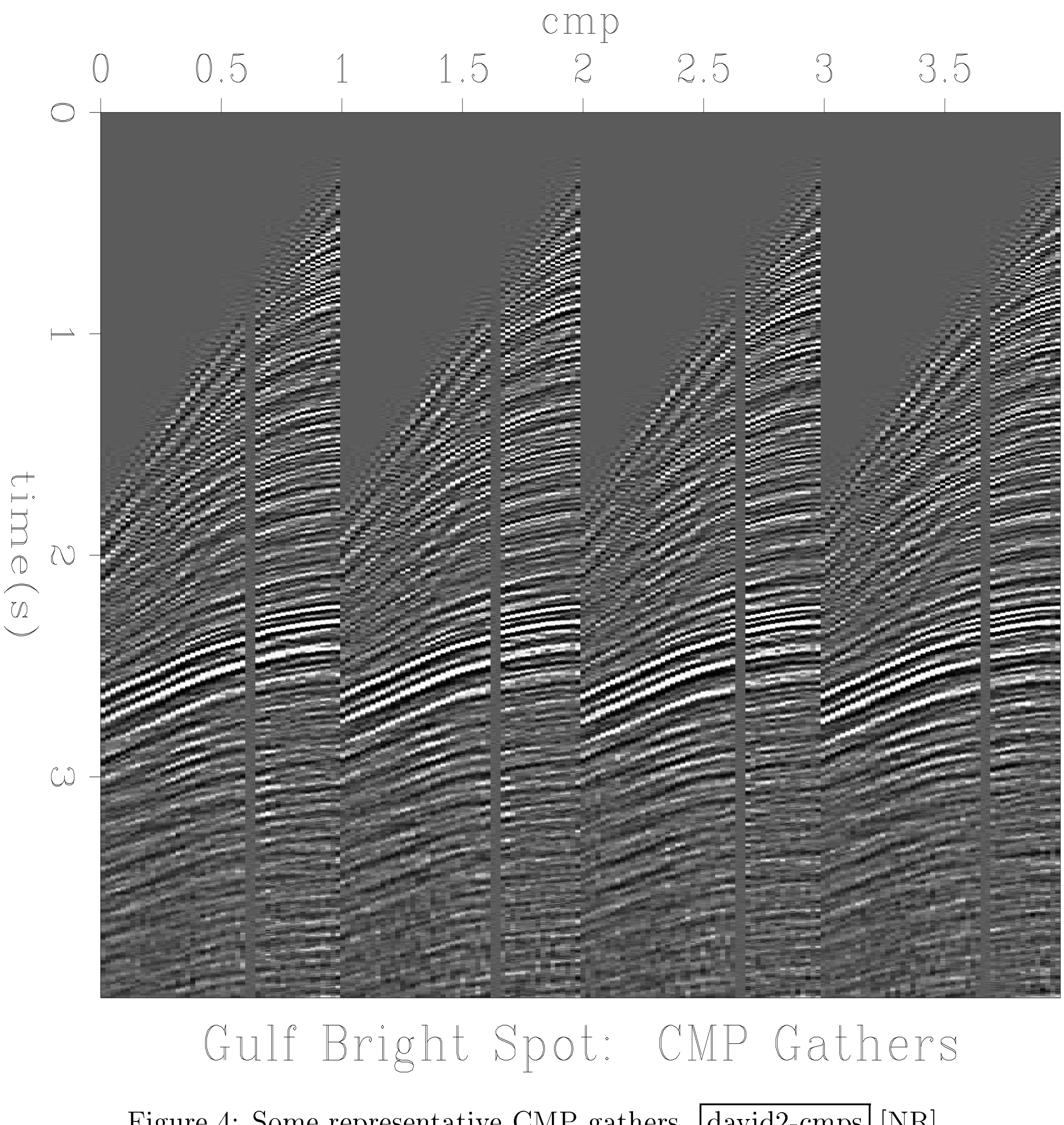

Figure 4: Some representative CMP gathers. $\boxed{\text{david2-cmps}}$ [NR]

Stacking velocity analysis

Figure **??** shows two stacking velocity semblance panels. The top panel corresponds to the CMP location at $x = 4$ km, and the bottom panel to $x = 4.7$ km. Figure **??** contains the same information, but is plotted in contours as opposed to raster. I have spent some effort trying to fine-tune the stacking velocity algorithm, to "tighten" the velocity spectrum, but perhaps some more work could be done as indicated by the residual semblance smear.

Currently, the main computational core of the CM code is running at a modest 120 Mflop/s on 8k processors. This is because the indirect memory addressing, shifting, and integer/boolean operations cumulatively require a large percentage of the cpu time compared to the low flop count. A stacking velocity analysis of 32 48-fold CMP gathers, 2,000 samples per trace, with 48 stacking velocities and 256 output time samples, requires a total of about 75 cpu seconds on 4k processors. About 30 s of that time is required to read input data from an NFS mounted disk to the front-end memory, another 10 s to transfer the data from the front-end to the CM, about 30 s to do the computations within the CM, and about 5 seconds to write the results from the CM to front-end to disk. Evidently, this process is heavily I/O bound, since I/O requires about 60% of the total run time. With a 10 Mb/s data vault or disk array, the I/O bottleneck could largely be eliminated (since the 32 gathers represent 12 Mb of data), which would speed up the effective run time by a factor of about 2.5x.

Migration velocity analysis

Figure **??** shows two migration velocity semblance panels. The top panel corresponds to the surface location at $x = 4$ km, and the bottom panel to $x = 4.7$ km. Figure **??** contains the same information, but is plotted in contours as opposed to raster. At this point, I have not spent any extra effort trying to fine-tune the migration velocity algorithm. Already, however, the migration velocity spectrum looks better resolved than the stacking velocity spectrum, especially when comparing the contour plots of Figure **??** to Figure **??**. Still, there is considerable migration semblance smear, and I need to fine-tune the algorithm.

Currently, the main computational core of the CM code is running at about 420 Mflop/s on 8k processors. A prestack migration velocity analysis of 32 48-fold CMP gathers, 2,000 samples per trace, with 48 stacking velocities and 256 output time samples, requires a total of about 255 cpu seconds on 4k processors. About 30 s of that time is required to read input data from an NFS mounted disk to the front-end memory, another 10 s to transfer the data from the front-end to the CM, about 210 s to do the computations within the CM, and about 5 seconds to write the results from the CM to front-end to disk. As opposed to the stacking velocity analysis algorithm, the migration velocity analysis is mostly cpu bound, since I/O requires only about 18% of the total run time. Addition of a data vault or fast disk array will only speed the migration velocity analysis by a factor of about 1.2x.

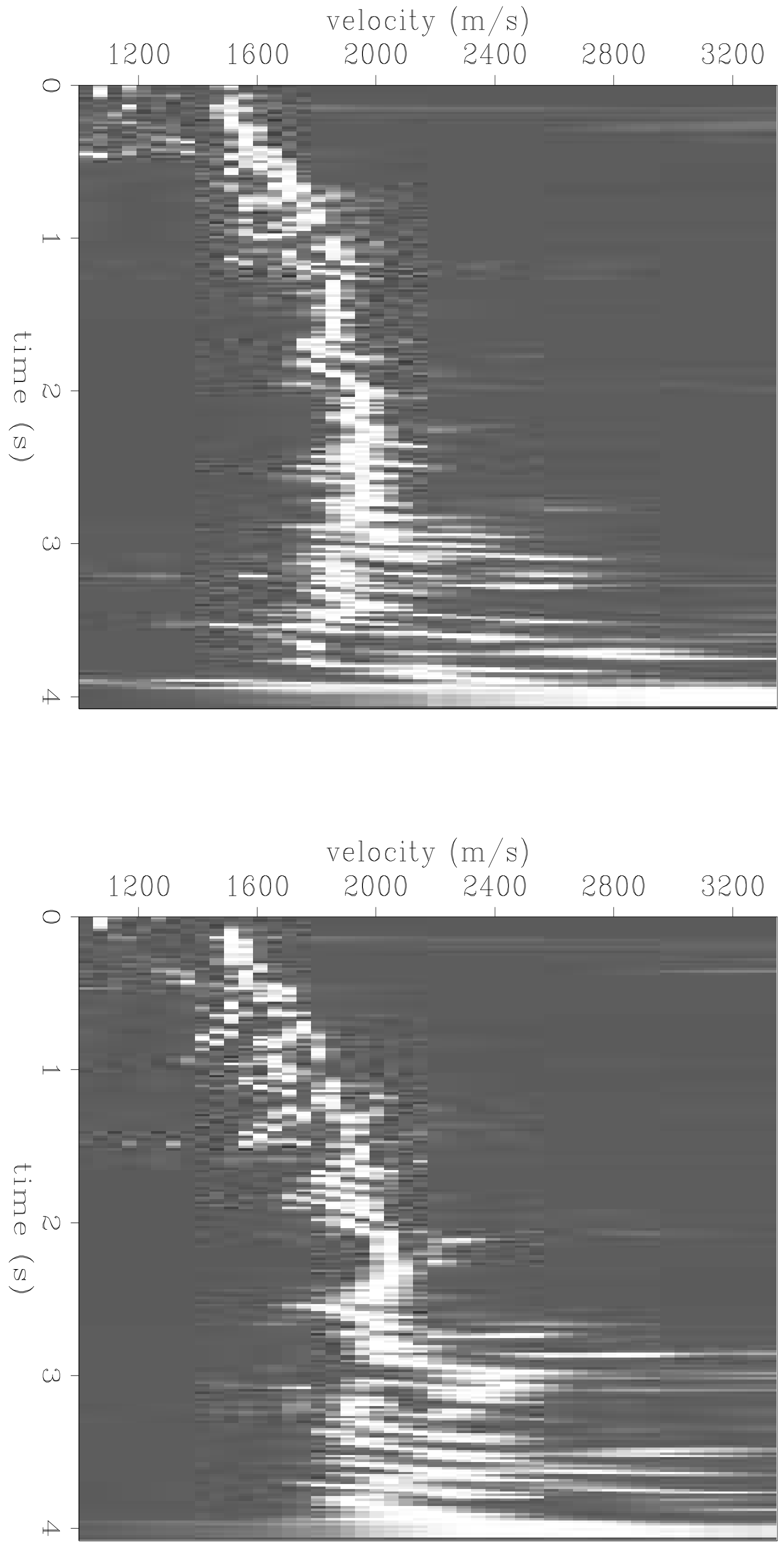

Figure 5: Stacking velocity panels, raster display. The top panel corresponds to a CMP location at 4.0 km, and the bottom panel corresponds to a CMP location at 4.7 km. david2-vstackR [NR]

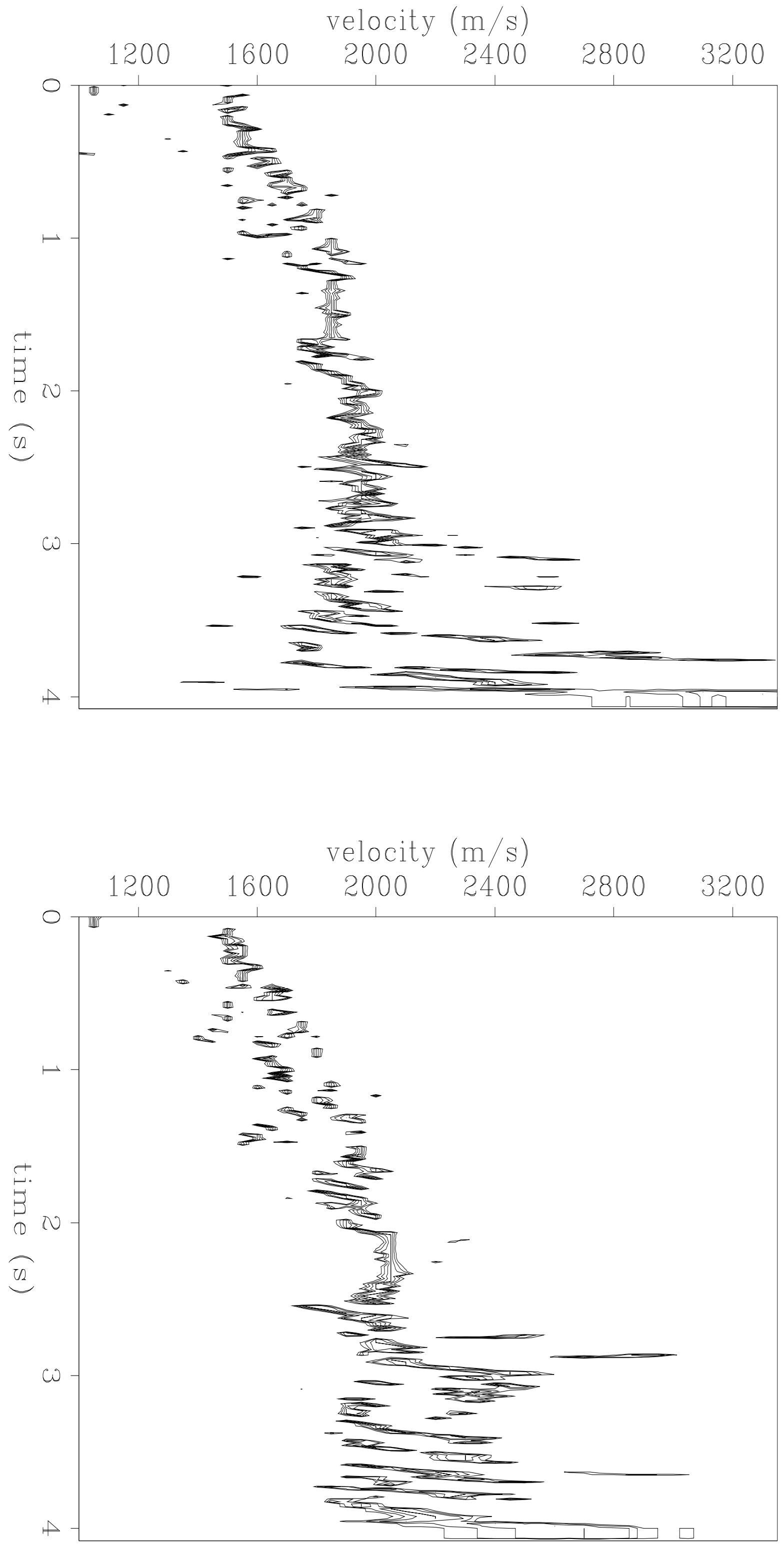

Figure 6: Stacking velocity panels, contour display. The top panel corresponds to a CMP location at 4.0 km, and the bottom panel corresponds to a CMP location at 4.7 km. $\boxed{\text{david2-vstackC}}$ [NR]

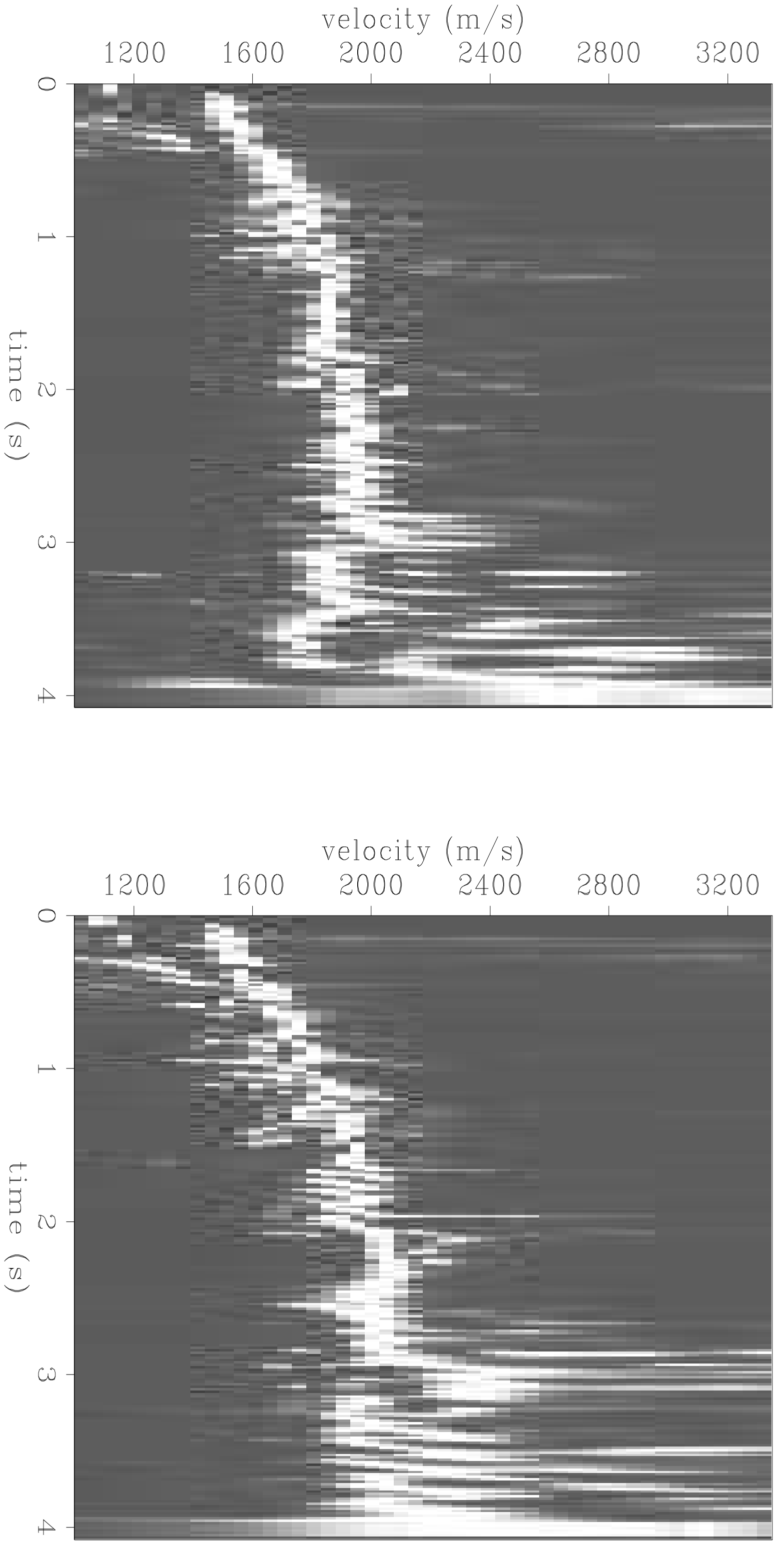

Figure 7: Migration velocity panels, raster display. The top panel corresponds to a surface location at 4.0 km, and the bottom panel corresponds to a surface location at 4.7 km. david2-ktvelR [NR]

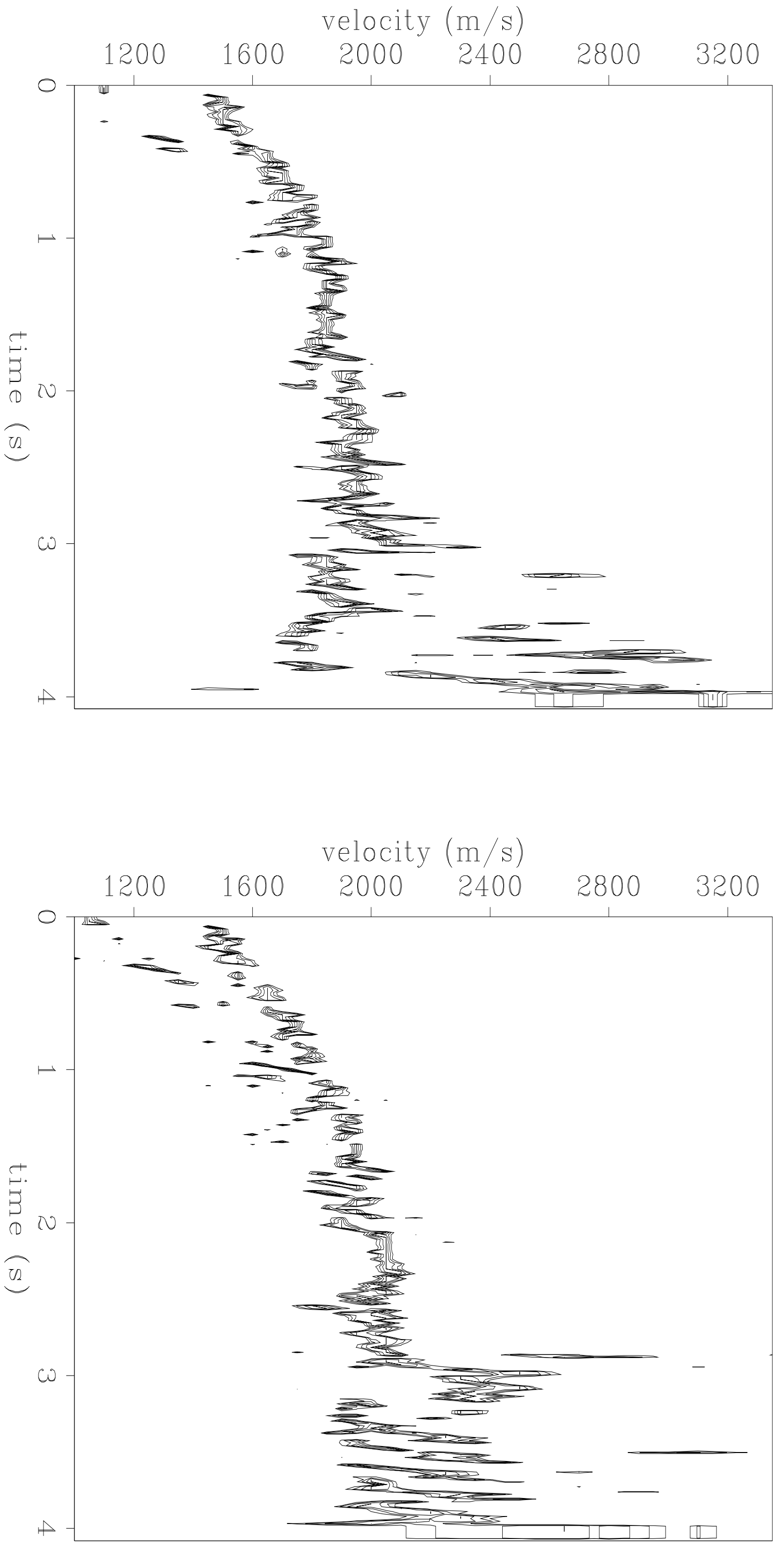

Figure 8: Migration velocity panels, contour display. The top panel corresponds to a surface location at 4.0 km, and the bottom panel corresponds to a surface location at 4.7 km. $\boxed{\text{david2-ktvelC}}$ [NR]

A brief interpretation

Comparison of the top semblance analysis to the bottom semblance analysis in Figure **??** demonstrates the potential for high-resolution migration velocity analyses in reservoir applications. The gas reservoir is located at about 2.25 s pseudo-depth in the near offset section of Figure **??**. The change in amplitude along the gas event from the surface location at 4.0 km to 4.7 km (700 m separation), from bright (gas?) to dim (no gas?), may be indicative of change in reservoir sand porosity and/or pore gas saturation. The velocity analyses in Figure **??** imply a clear change in kinematic hyperbolic moveout spatially along the reservoir, since the migration velocity at reservoir depth is about 1950 m/s at 4.0 km (gas?) and 2050 m/s at 4.7 km (no gas?). This change of 100 m/s in migration velocity implies a much larger change in interval velocity along the reservoir, on the order of 500 m/s, which is consistent with sandstone compressional velocities under dry versus saturated conditions (Lumley et al., 1992). Of course, any responsible interpretation of such velocity analysis results should include correlation with in situ borehole measurements.

Data example summary

In summary, this data example clearly demonstrates the potential for enhanced reservoir characterization by high-resolution surface seismic velocity analyses. The stacking velocity analysis took about 1 cpu second per CMP gather on 4k processors, and about 7 cpu seconds per semblance panel for the migration velocity analysis, not including I/O times. The stacking velocity analysis is heavily I/O bound, but could be made faster by a factor of about 2.5x with the use of a 10 Mb/s data vault or fast disk array. The migration velocity analysis is mostly cpu bound, and could only be marginally improved by about 20% with fast I/O.

CONCLUSIONS

Three-dimensional prestack seismic reservoir characterization may benefit from high resolution velocity analyses. The intensive computational effort required to perform 3-D velocity analysis requires implementation on a supercomputer. I have developed and implemented two velocity analysis algorithms on a massively parallel Connection Machine. The first algorithm performs stacking velocity semblance analysis, and tends to be I/O bound. The second algorithm performs prestack time migration velocity semblance analysis, and tends to be cpu bound. The inner core computational loop of the stacking velocity analysis code runs at about 120 Mflop/s, whereas the migration velocity analysis code runs at about 420 Mflop/s, on an 8k processor CM-2. On a full size 64k CM-2, the stacking velocity code scales to about 1 Gflop/s, and the migration velocity code scales to about 3 Gflop/s. These rates are fast enough to make high-resolution 3-D velocity analyses feasible in an applications research environment.

ACKNOWLEDGMENTS

I would like to thank Ray Abma, Biondo Biondi, Dave Nichols, and Lin Zhang for their constructive comments. I am also grateful to Thinking Machines Corporation, the SEP Supercomputing Tier, and Jon Claerbout for supporting the supercomputing facility here at SEP, necessary for work in 3-D prestack seismic reservoir characterization.

REFERENCES

Han, D., 1986, Effects of porosity and clay content on acoustic properties of sandstones and unconsolidated sediments: Ph.D. thesis, Stanford University.

Lumley, D., Bevc, D., Ji, J., and Tálas, T., 1992, Lab measurement of elastic velocities in dry and saturated Massillon sandstone: Stanford Exploration Project Report, **73**, Stanford University.

Lumley, D. E., 1991, Some aspirations in seismic reservoir characterization: Stanford Exploration Project Report, **70**, Stanford University, 161–164.

Lumley, D. E., and Beydoun, W. B., 1991, Elastic parameter estimation by Kirchhoff prestack depth migration/inversion: Stanford Exploration Project Report, **70**, Stanford University, 165–192.

Lumley, D. E., and Biondi, B., 1991, Kirchhoff 3-D prestack time migration on the Connection Machine: Stanford Exploration Project Report, **72**, Stanford University, 123–133.

Parsons, R., 1986, Estimating reservoir mechanical properties using constant offset images of reflection coefficients and incident angles: 56th Ann. Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts, 617–620.

SEP POT LUCK BAR-B-QUE

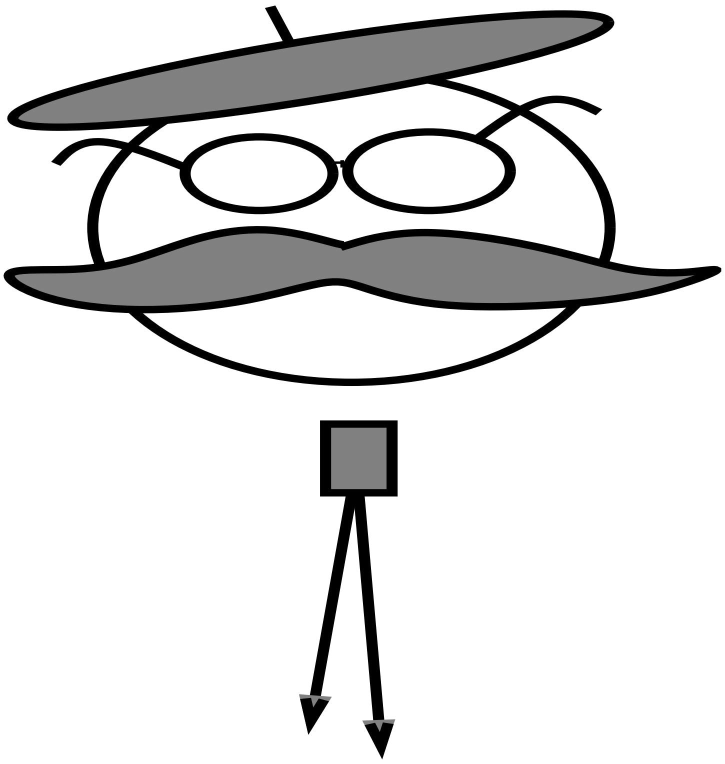

Figure 9: This caricature is in no way intended to represent any famous geophysicist, whether factual or fictional, nor is it intended to be misconstrued as an official endorsement of the event in question by the Stanford Exploration Project or the Regents of Stanford University. david2-jon [NR]

```
WHO:    all SEP members + their guests
        (we will be "hanging around" the BBQ in the back yard)

WHEN:   Saturday April 25th @ 3pm -->

WHERE:  David and Mihai's residences
        58B--58C Escondido Village

WHY:    If you have to ask then get back to the $%&*@# office!

WHAT:   You need to bring:
        * your own BBQ meat and booze
        * one of: a salad, a vegetable dish, a dessert, etc.

        We will supply the BBQs, chips and soft drinks...
```