Velocity analysis by prestack migration

Kamal Al-Yahya

INTRODUCTION

This paper is a continuation to the discussion on velocity analysis by prestack migration in Al-Yahya and Muir (1984). Some references will be made to earlier work; however, an effort has been made to make this paper self-contained. The reader may remember that the method uses prestack migration results (profile migration to be specific) to estimate the velocity (or residual velocity). Before starting the discussion, I will address some issues regarding the philosophy of the method.

Why *migration*?

The conventional velocity analysis, which uses NMO, relies on horizontal reflectors. In the case of a plane dipping reflector, the stacking velocity is higher than the root-mean-square, *rms*, velocity (which is what the NMO equation gives) by a factor equal to the secant of the dip angle (Levin, 1971). When diffracters exist, they cannot be represented by a single dip and in general when the structure is complex, the NMO approach is invalid.

To make velocity analysis more general, NMO is replaced by migration. Migration has the known property of collapsing a diffraction to a focus. Therefore, the method will work even if the seismic events are not reflections. Since diffraction is the most fundamental seismic event, the method is *structure independent* and does not rely on horizontal reflectors.

Why *prestack* migration?

Travel-time versus offset serves as a velocity indicator and therefore the velocity can be determined by any function that relates travel time, offset, and velocity. In this respect, the NMO equation is only one possibility; migration is another. Keeping the offset in the formulation requires that the migration be done prior to stacking. Post-stack migration velocity analysis (as in Rothman, 1983) will necessarily give a velocity that is a function of the stacking velocity.

Why *profile* migration?

Of the various pre-stack migration methods, I have selected profile migration as suggested by Jacobs (1982). He proposed profile migration because sampling of the geophone axis is better than that of the shot or offset axes, so spatial aliasing is reduced. Another advantage of profile migration is that profiles are physical experiments and can be treated independently. Also, since the proposed method relies on examining Common Receiver Gathers (CRG's), the shot-geophone (s, g) space becomes a natural choice. (Note that migration schemes other than profile migration exist in the (s, g) space; see Thorson (1980) for example, where common-frequency slices are used.)

The examples in this paper were computed in the frequency domain, although it is possible to work in the time domain. When migrating in the (s, g) space, I used the 45-degree finite difference equation. The appendix shows an algorithm for profile migration in the frequency-wavenumber domain. I sometimes used it when migrating with a constant velocity.

Parallelism

Concern is often raised about the cost of prestack migration. Since in prestack migration velocity analysis the process is repeated, cost is a more serious concern. No claim is made that the method competes in speed with conventional velocity analysis. It should be pointed out, however, that the cost is decreasing as more powerful computers, in terms of both speed and parallelism, are introduced.

As illustrated in Figure 1, there is a high degree of parallelism in profile migration. The first parallelism is over the shot axis. Since I do the migration in the frequency domain, there is also another parallelism over the frequency axis. When extrapolating the data from one depth to the next, frequencies are independent and can therefore be extrapolated in parallel. The third parallelism is over offset. If finite differencing is used, it is not a complete parallelism since neighboring points are related to one another; many

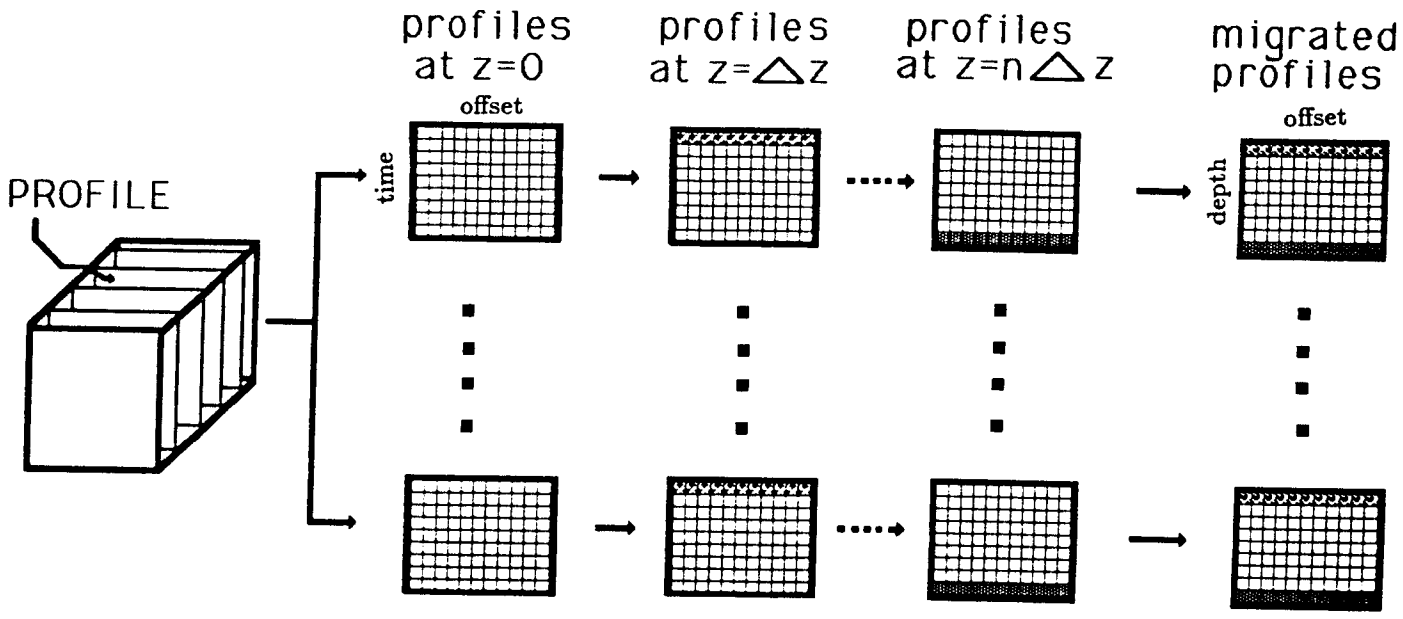


FIG. 1. Profile migration scheme. The shaded strip is the image at the given depth at t=0. The strip is placed in the appropriate position in the migrated profile.

loops over offset can however be computed in parallel. If the extrapolation is done in the wave-number domain the parallelism is complete.

In a machine where parallelism is not possible, all the parallelism is lost. The opposite extreme is to have all parallelism available. Full parallelism is possible only if we can bring all profiles to the virtual memory of the computer, or send each profile to a different processor, both of which do not seem to be easily possible and accessible in the near future. However, parallel processors between those two extremes are becoming increasingly abundant. In a vectorizing machine only one parallelism can be taken advantage of, normally over the offset. Even with this limited parallelism, it has been possible to migrate profiles fairly quickly. It will not be long before we see *matrix-operation* machines that will make it possible to migrate a profile in a matter of few CPU seconds, substantially lowering the cost of profile migration so that it is cheap enough to do on a routine basis and can be used in velocity analysis.

THE PRINCIPLE

The principle behind prestack migration velocity analysis is that **only when imaged correctly will an event be horizontally aligned in a migrated Common Receiver Gather (CRG).** The interpretation of this statement is that the image of the subsurface should be in the same location regardless of the source position. The parallel concept in a Common Midpoint Gather (CMP) is the alignment of reflections after an NMO with the appropriate velocity. In NMO, the "appropriate" velocity is the *rms* velocity. If we restrict velocity variation to depth only, we can also use rms velocity in migration velocity analysis. This restriction is, however, not necessary.

Utilizing the above principle, two approaches can be taken to determine the velocity. The first approach is iterative. That is, after migrating with an initial velocity, we attempt to quantify the error in the velocity function in our first trial. The advantage of this approach is economy where it is expected that only few iterations (say 3 or 4) will be needed. The quantification of the error may not be exact, but in each iteration it brings us closer to the correct velocity function. Each time we migrate the data, we obtain information about how far (or close) we are to the medium velocity.

The second approach is literally using the horizontal alignment principle; we simply scan the velocity space and subsequently check which velocity function satisfies the principle. This approach is more expensive than the iterative method. However, because it is a potential method, I will discuss it in a later section.

THE ITERATIVE METHOD

To update our velocity function, we need a way of quantifying the deviation of the events in a CRG from horizontal alignment. Let us see what happens if we migrate the data with a constant velocity v_m which is not the same as the medium constant velocity v. The migration involves downward continuing the source from the surface to depth z_m, followed by a time shift equal to $\sqrt{x^2+z_m^2}/v_m$, where x is the source-receiver separation. The travel time, t, associated with these two steps (downward continuation and time shifting), is then

$$t = \frac{z_m}{v_m} + \frac{\sqrt{x^2+z_m^2}}{v_m} \quad , \tag{1}$$

while if we used the medium velocity we have

$$t = \frac{z}{v} + \frac{\sqrt{x^2+z^2}}{v} \quad . \tag{2}$$

Eliminating t between equations (1) and (2), we obtain

$$z_m = \gamma z + \frac{x^2(\gamma^2 - 1)}{2\gamma(z + \sqrt{x^2 + z^2})} \quad , \tag{3}$$

where $\gamma = v_m/v$.

The first term on the right in equation (3) is a shift at zero offset; the second term is a moveout term. Since we want to determine γ from the moveout, we want to get rid of the shift. This is done by recasting equation (3) in time

$$\tau_m = \tau + \frac{x^2(\gamma^2 - 1)}{2(v_m^2\tau + v_m\sqrt{\gamma^2 x^2 + v_m^2\tau^2})} \quad , \tag{4}$$

where $\tau = z/v$ and $\tau_m = z_m/v_m$. Using time instead of depth will also prove useful in the search method to be discussed later. The basic advantage is that the event of interest stays in the same τ level, rather than at various z levels. When migration results are expressed in depth, equation (3) will be used. In those cases γz will be used and therefore, limiting all combinations of γ and z that produce the same z_m to the same level.

Equation (4) gives a relation between the apparent travel time and the actual travel time, which should be equal regardless of offset when the migration velocity is equal to the medium velocity (γ=1). Note also that at zero offset ($x = 0$), the apparent travel time is equal to the true travel time regardless of migration velocity; error in velocity is manifested only in non-zero offsets. After migrating with a guessed velocity, we can search CRG's to see what trajectories give coherent events and thereby determine γ, and subsequently the medium velocity, $v = v_m/\gamma$. Such a search is similar to the search done in conventional velocity analysis, the difference being that here we search for a velocity ratio, γ, instead of velocity.

Note that the derivation was based on constant velocity. As the examples will show, the γ value determined for a layer is of course related to all γ values above that layer. For a depth-dependent velocity, we can calculate the interval velocity from the value of γ by a method similar to Dix equation. For a general velocity variation, a tomographic correction to the model is applicable. If determination of these γ values is perfect, only one iteration is needed. It is expected that more than one iteration will be needed.

Example 1

Let's use equation (4) for the case of a reflector dipping 30° in a constant velocity medium of 1.5 km/sec. Three (separate) migrations were done; one with the correct velocity, another with a lower velocity (1 km/sec), and another with a higher velocity

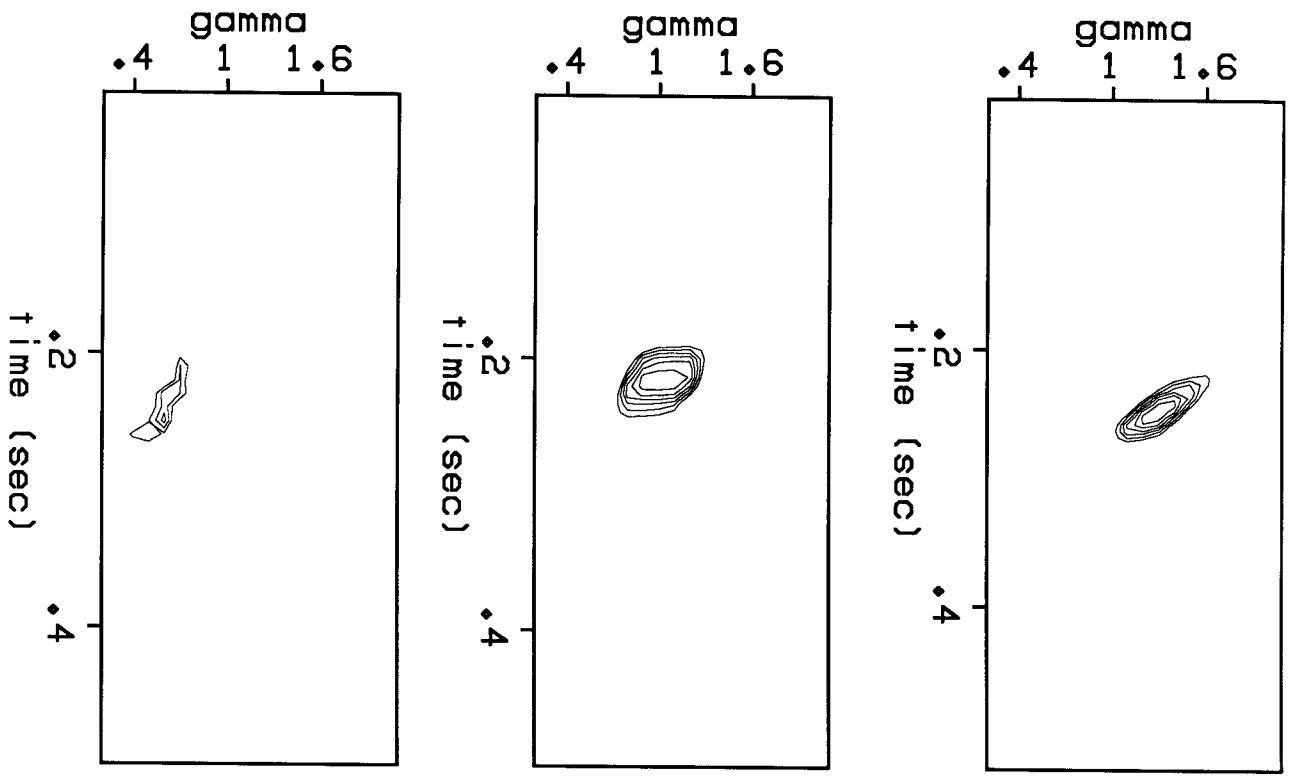


FIG. 2. Migration velocity analysis for a migrated CRG over a dipping reflector. Left: using a low velocity (1 km/sec). Center: using the correct velocity (1.5 km/sec). Right: using a high velocity (1.9 km/sec).

(1.9 km/sec). A similar example was used in Al-Yahya and Muir (1984); profiles, their migration, and CRG's are shown there (for two velocities) and will not be repeated here. Migration velocity analysis panels for a CRG are shown in Figure 2 for each velocity. The figure shows that γ=1 for the correct velocity while γ=.7 for the low velocity and γ=1.3 for the high velocity, which are close to the actual γ values of 1, .667, and 1.267 respectively. Had we, say, migrated the profiles with the low velocity, the velocity analysis would indicate that we should use a higher velocity in the subsequent migration.

This example is very simplistic in that the velocity is characterized by only a single parameter. The next step towards complexity is taken in the next example.

Example 2

Now we apply the method on a three-layer model. The layers have velocities from top to bottom of 2, 2.5, and 3 km/sec; the depth of boundaries are .3, 1, 1.8 km, respectively. A synthetic profile for this model (generated by Pete Mora's elastic modeling program) is shown in Figure 3. The profiles were migrated with a constant velocity of 2 km/sec, and the result is shown in Figure 4. Figure 5 shows velocity analysis using equation (3)

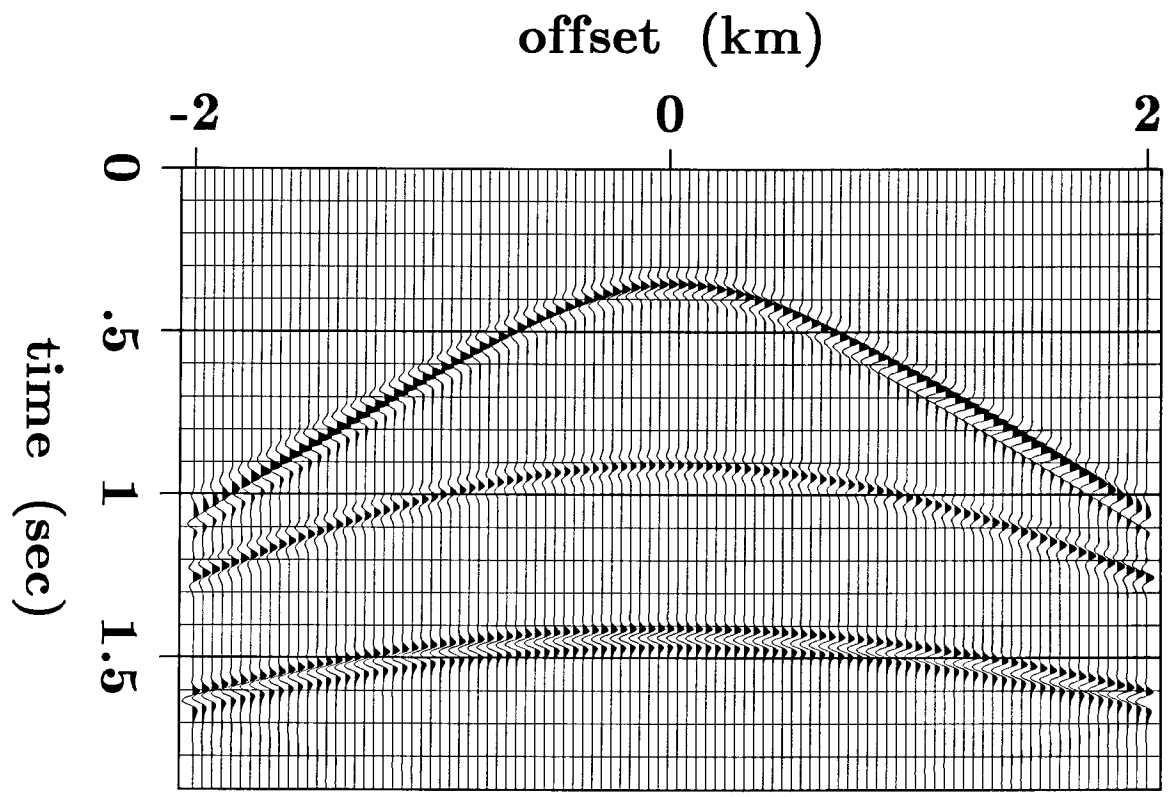


FIG. 3. Synthetic profile for a three-layer model. The velocities of the layers from top to bottom are 2, 2.5, and 3 km/sec. The velocity of the lower half space is 2 km/sec.

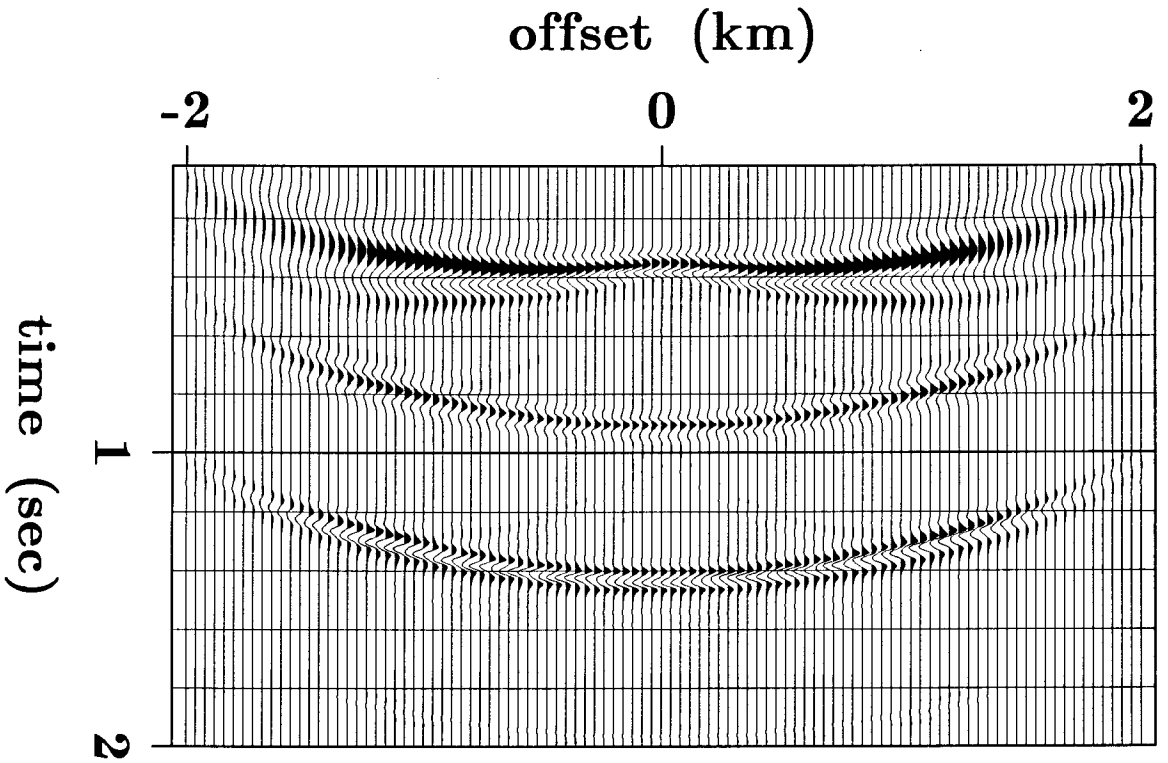


FIG. 4. The result of migrating the profile of Figure 3 with a constant velocity of 2 km/sec.

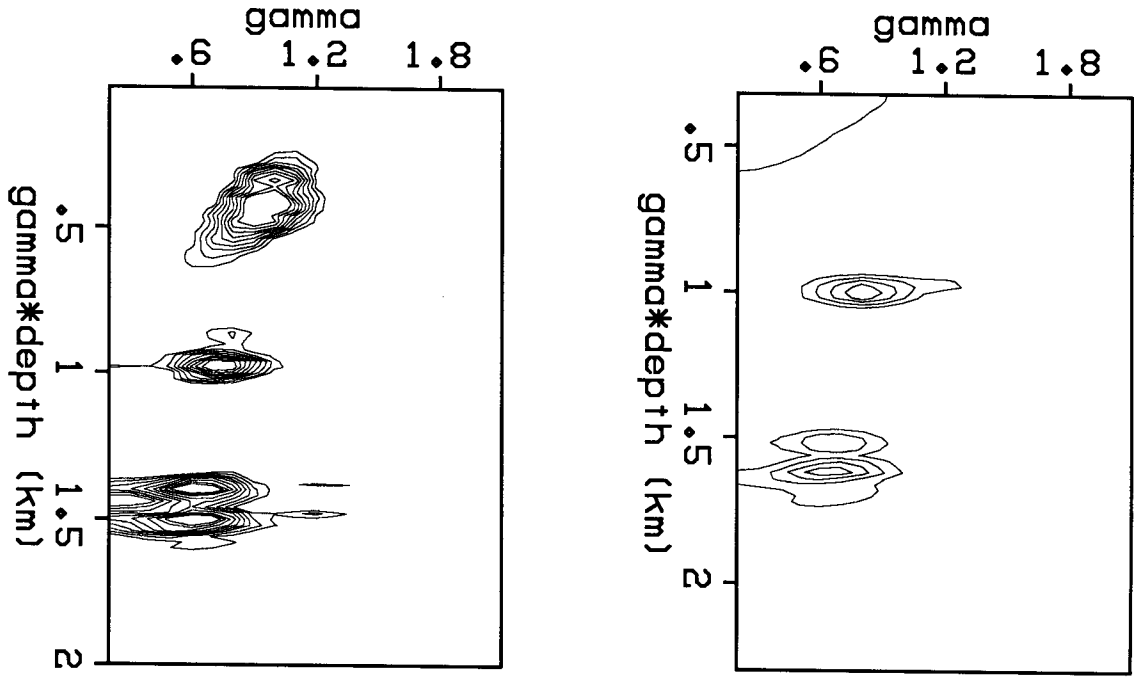


FIG. 5. Left: Migration velocity analysis for a migrated CRG from Figure 4. Right: Migration velocity analysis for the same CRG after stripping the first layer.

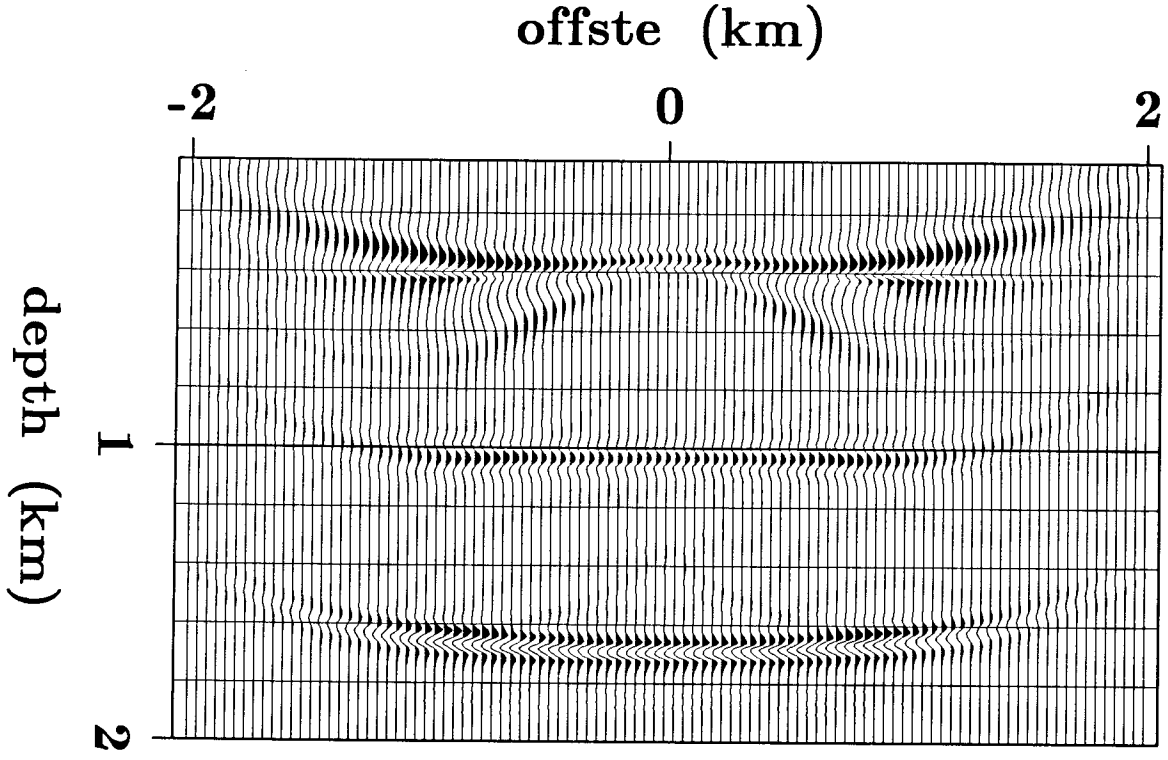


FIG. 6. The result of migrating the profile of Figure 3 using the correct velocity for the top two layers only.

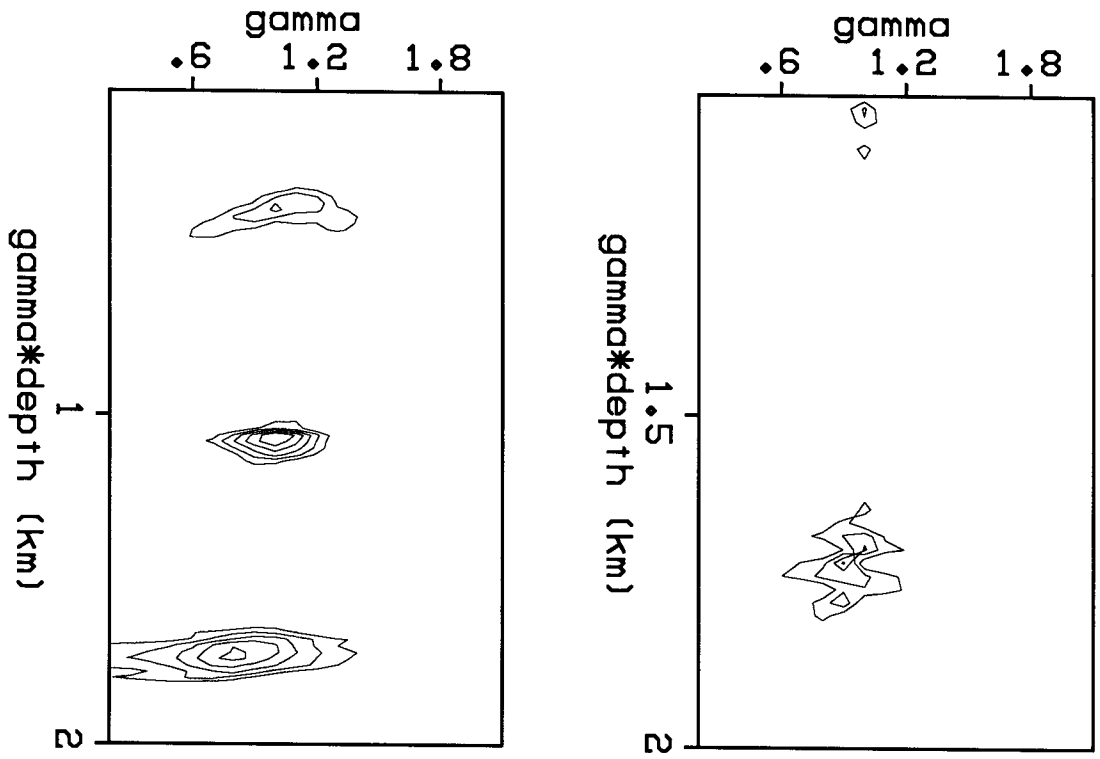


FIG. 7. Migration velocity analysis for the CRG of Figure 6. Right: Migration velocity analysis after stripping the top two layer.

on a migrated CRG (note that the vertical axis is labeled $\gamma * depth$ as mentioned in the discussion of equation (3)). We see that only the first layer was migrated with γ=1 (the correct velocity). The figure shows that the other layers were migrated with low velocities, and therefore in the next pass of migration we should increase the velocities using γ values obtained from the velocity analysis. The error in the velocity of the second layer involves only that layer, since the first layer was migrated correctly.

Alternately, we can exclude the correctly imaged region by *stripping* it. Figure 5 shows the velocity analysis after stripping the top layer. γ for the second layer is now determined to be .8, meaning that the velocity of the second layer is 2/.8=2.5 km/sec. Figure 6 shows the result of migrating the profiles with the updated velocity function and Figure 7 shows the corresponding velocity analysis. Again, we can strip the top two layers and determine the residual velocity for the third layer, as shown in Figure 7. Finally the profiles are migrated with the correct velocity for the top three layers (velocity of the half space has no effect on the layers above it). The migrated profile is shown in Figure 8 along with the corresponding velocity analysis.

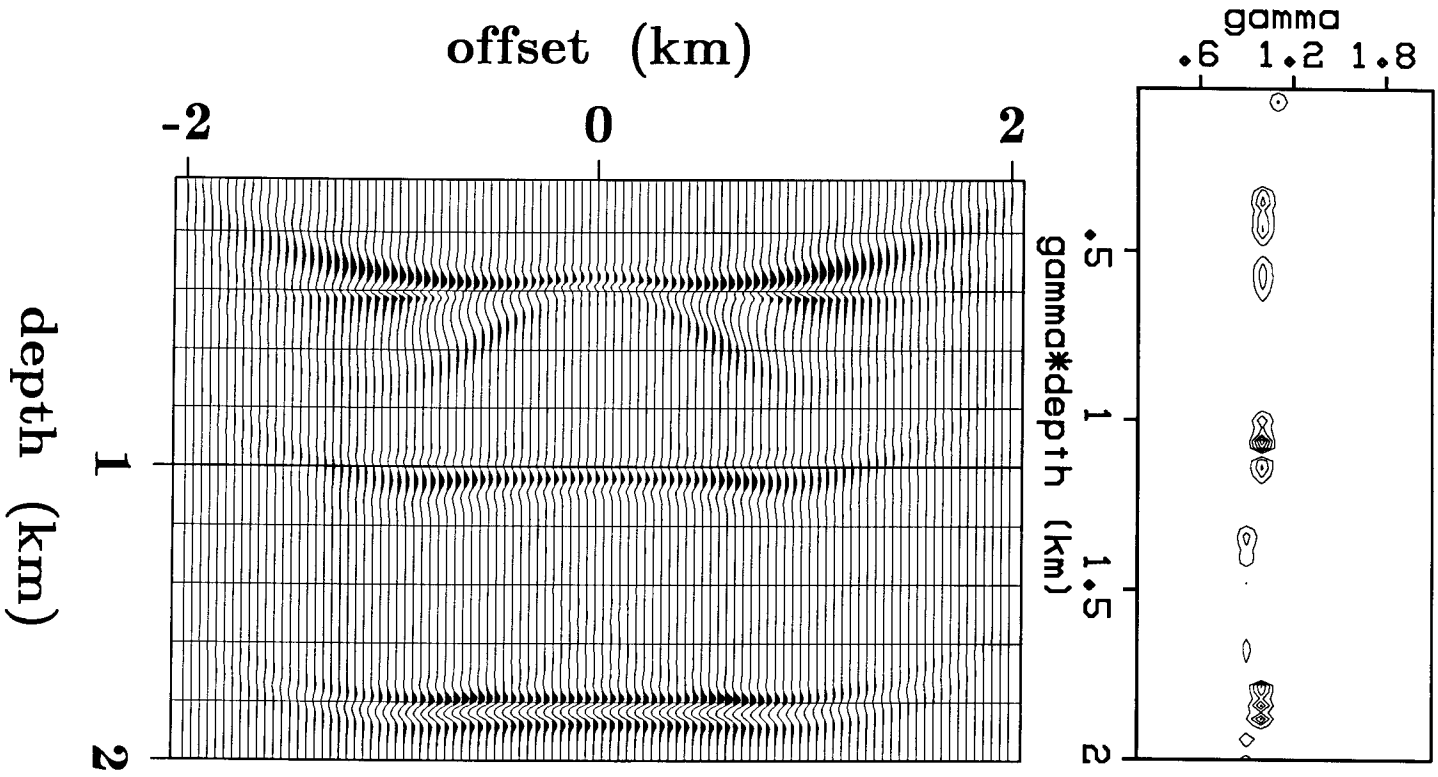


FIG. 8. Left: the result of migrating the profile of Figure 3 using the correct velocity for the top three layers. Right: migration velocity analysis of the CRG.

THE SEARCH METHOD

The main advantage of the iterative method is that it promises to bring us closer to the correct velocity in each iteration, which makes it more economical than the search method. On the other hand, in Al-Yahya and Muir (1984) the following statement was made:

> It should be noticed that, according to the principle stated above, the horizontal alignment of images in migrated common receiver gathers (CRG's) will take place only when the velocity is correct; this fact should be made use of.

This means that we can search the velocity space by migrating with several velocities to determine what velocity function aligns all events in the CRG's. Of course this implies a cost higher than that of the iterative method. With the advent of more powerful computers, I think it is worthwhile to try such a method.

To check the alignment, we can stack the CRG and see which velocity gives the best stack at each travel time. Stacking can be replaced by semblance or other measures. The

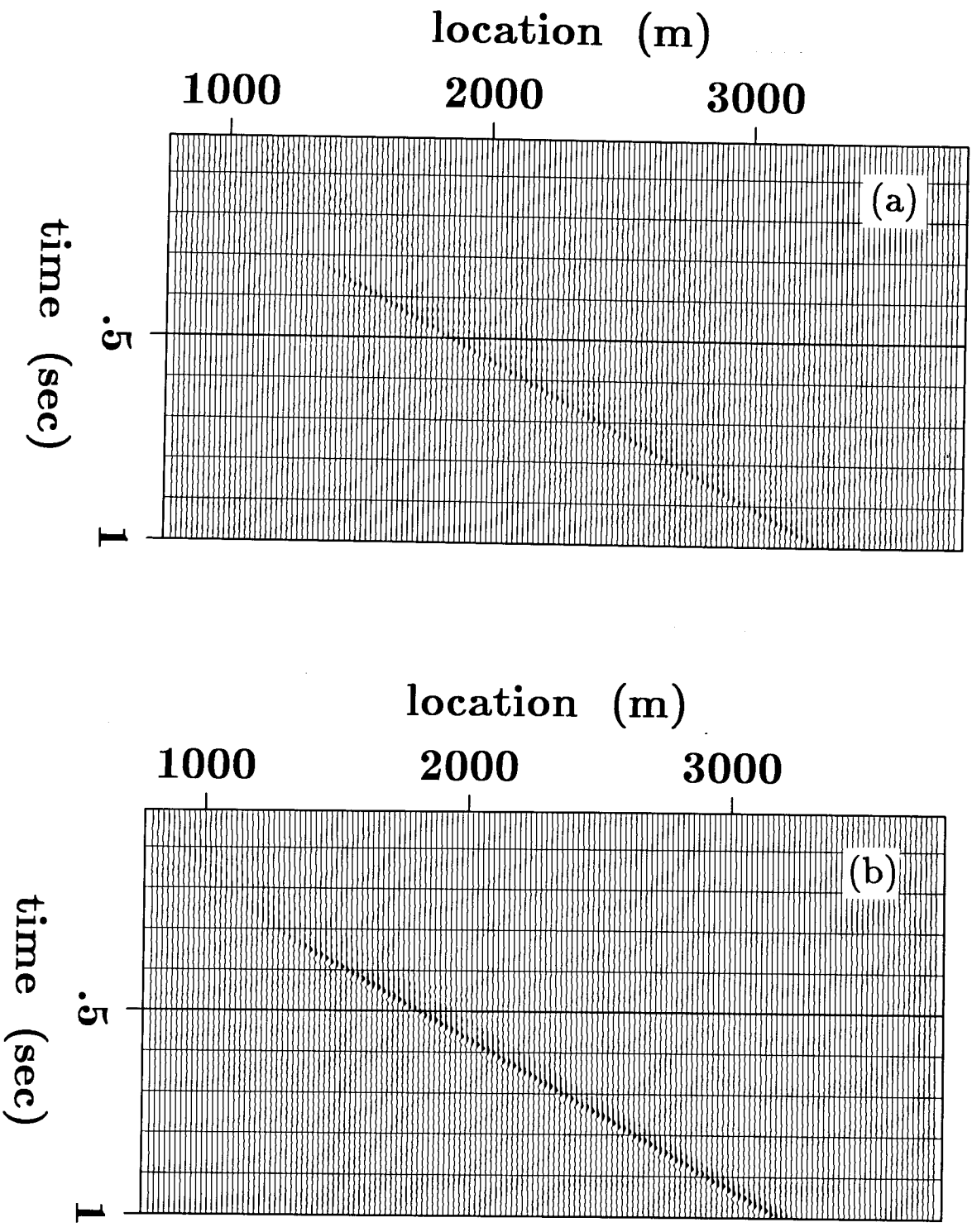
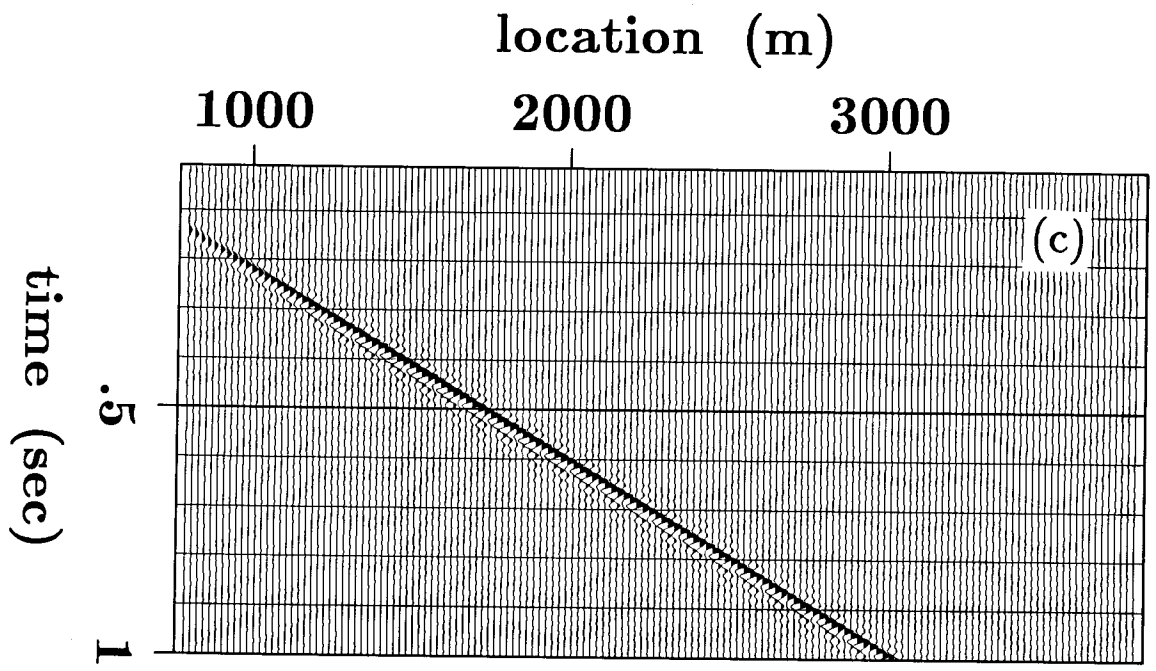


FIG. 9. The stack of profiles over a dipping reflector migrated with four constant velocities. (a) 1 km/sec. (b) 1.2 km/sec. (c) 1.5 km/sec. (d) 1.9 km/sec.



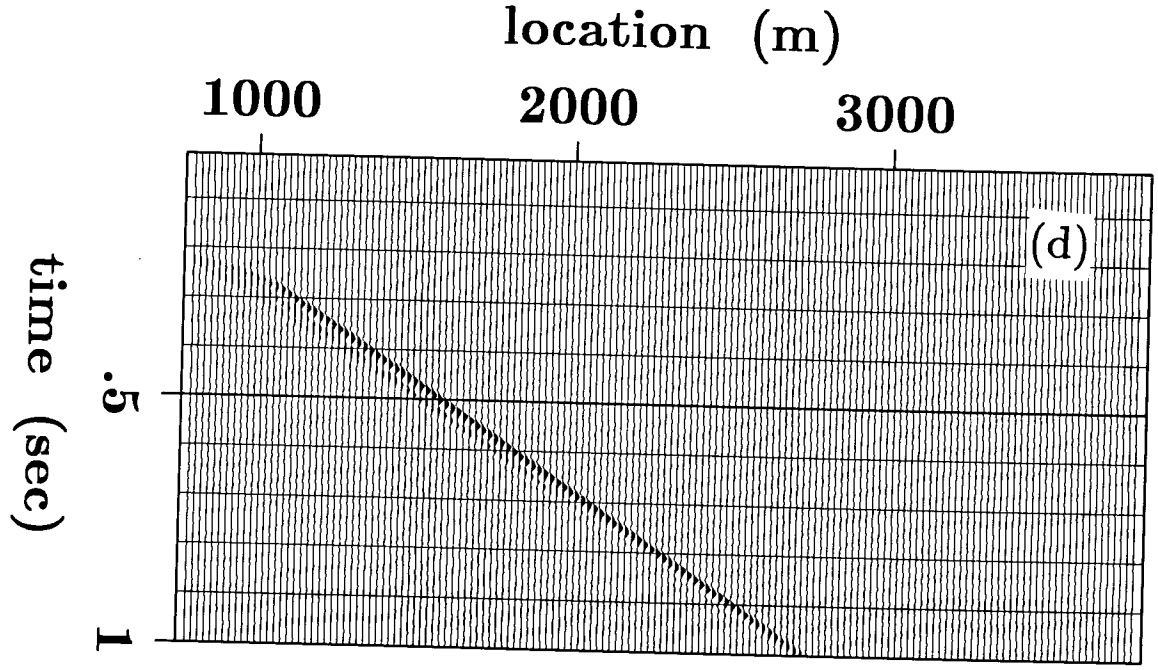


FIG. 9. continued

velocity space to be scanned is of course very large; it is prohibitively large. We can however use constant velocities which are sufficiently spaced. The interval velocities are obtained from the velocities used in migration by a method similar to Dix equation for layered media. Tomographic reconstruction can be used when velocity changes laterally as well. When constant-velocity migration is followed by stacking, the method closely resembles the constant-velocity NMO and stacking used in conventional velocity analysis. The cost of migration is however much larger than NMO. Fowler (1985) has also used constant velocity pre-stack Stolt migration, with a model-driven scheme (that is working from interval velocities) which is an extension of Toldi's method (1985) to the migration case.

The fact that the correct velocity aligns events in CRG's and therefore gives the best stack means that the power in the stack can be used as an objective function, just like what Rothman (1985) used for the statics problem. Had we migrated in the midpoint-offset (y, h) domain, a natural objective function would be the ratio of the power near the zero-offset to the power in the rest of the CMP gather.

Example 1

Let us see what happens if we apply the search method to the example of Figure 2. A total of 10 velocities were used; four of the stacks are shown in Figure 9. Figure 10 shows a stacked CRG as a function of migration velocity where we see that had we used depth instead of time, the event would lie in various z levels. The figure also shows that the stack is best at a velocity of 1.5 km/sec, which we can pick as the correct velocity. We can visually see the same thing in the stacks of Figure 9. Figure 11 quantifies this visual observation by showing the power of the stack as a function of migration velocity, which in this case represents samples of the objective function. Since the migration velocities are constant, each point in the abscissa can be made to represent a single parameter. In general, it should represent a *function* and the ordinate becomes a *functional.* Also, since there is only one parameter to determine in this example, there is only one maximum in the objective function with no local maxima.

Example 2

Now we return to the three-layer example that was shown in the iterative method. Figure 12 shows a migrated and stacked CRG for a suite of constant velocities and should be compared to the conventional velocity spectrum shown next to it. We see that the first large signal is at a velocity of 2 km/sec and at a travel time of .3 sec, which are the correct

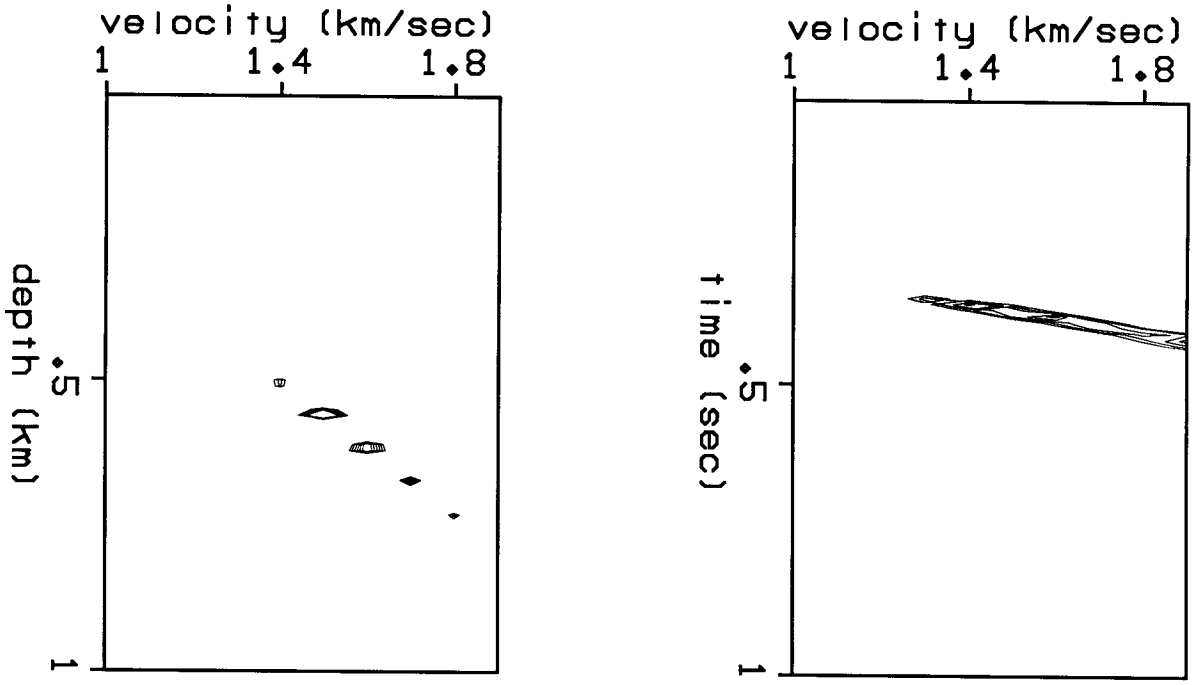


FIG. 10. The stack of a CRG over a dipping reflector as a function of velocity. Left: when depth is used. Right: when time is used.

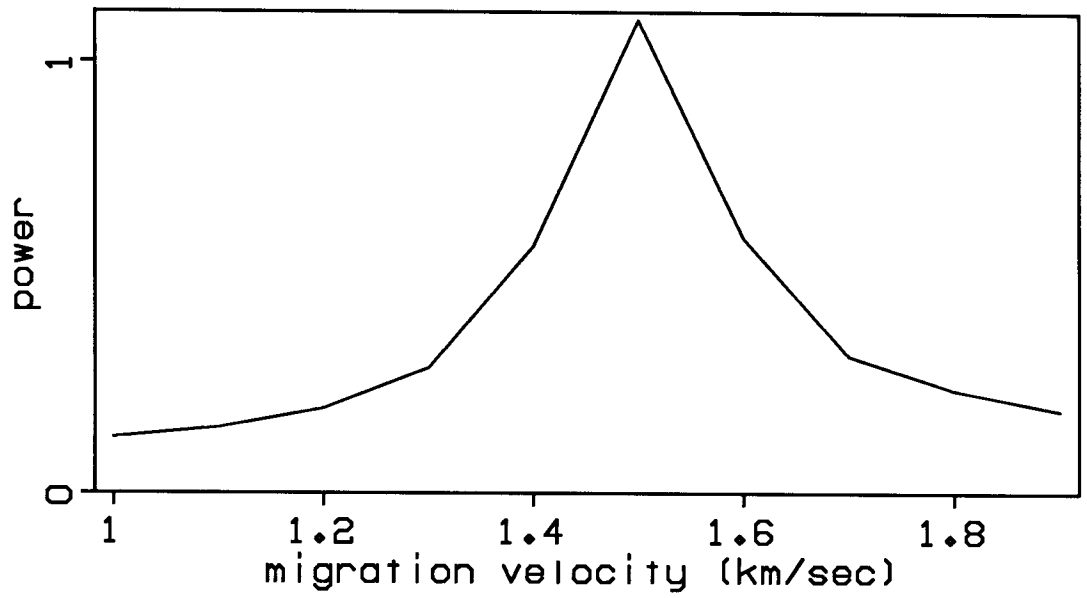


FIG. 11. Power of the stack as a function of migration velocity for the dipping reflector model.

values for the first layer. The second large signal is at 2 km/sec and about .43 sec. The velocity for the second layer is, however, 2.5 km/sec. The second reflector stacks well at a lower velocity because, as mentioned earlier, the error involves the previous layer. The same applies to the third reflector which stacks well for a velocity of 2.6 km/sec.

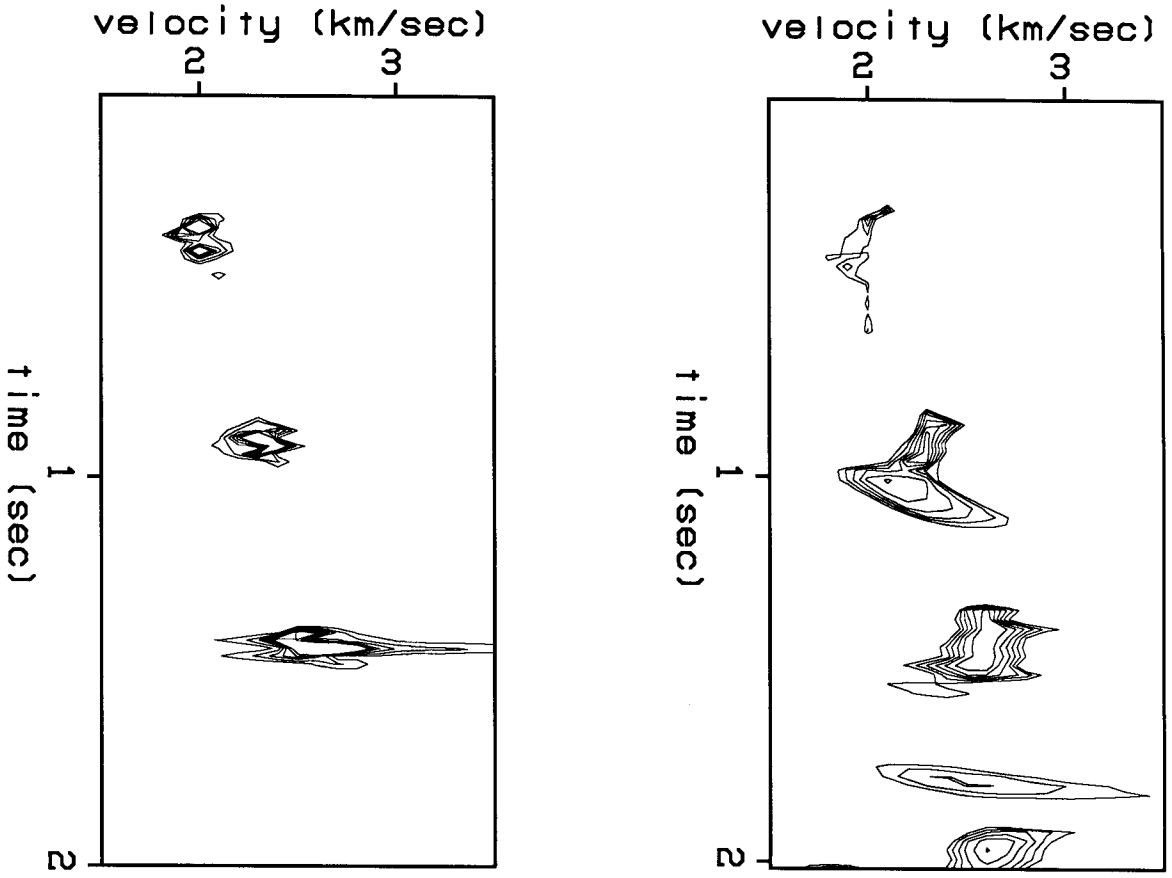


FIG. 12. Left: migrated and stacked CRG over the three-layer model. Right: conventional velocity spectrum over the same model.

Several velocity functions were used in this example. We have therefore probed the objective function and, out of curiousity, I plotted some of its available samples, shown in Figure 13. Note that it differs from Figure 11 in two ways. First, the abscissa represents *functions* and not values. Second, there are several maxima; the "global" maximum corresponds to the correct velocity used in Figure 8.

FIELD DATA

So far, all the examples shown have been synthetic. They have served to demonstrate the principle behind prestack migration velocity analysis. They have also shown

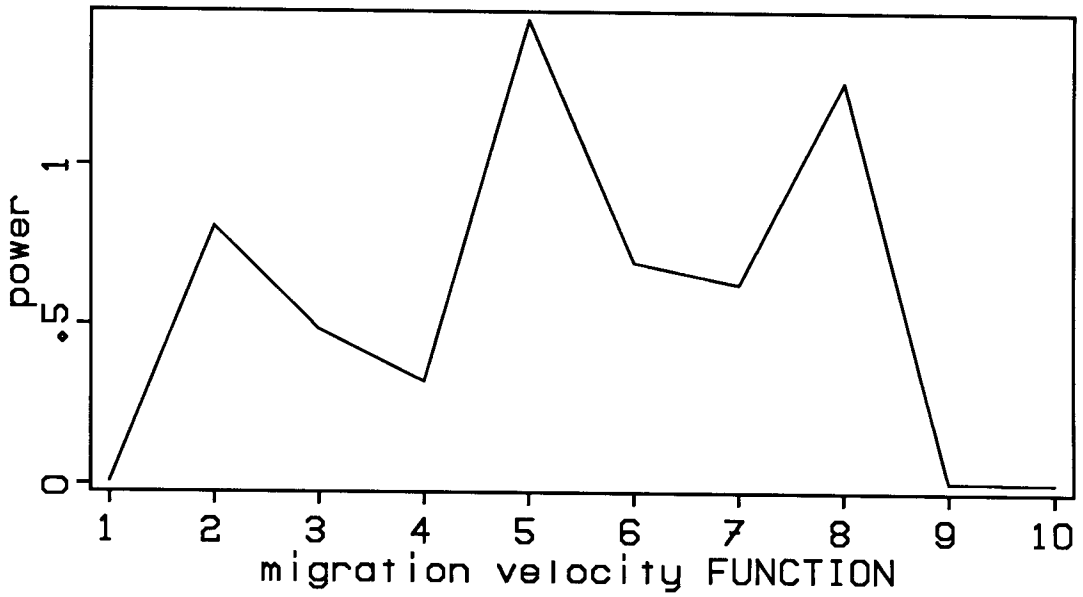


FIG. 13. Power of the stack as a function of migration velocity function for the model in Figure 3.

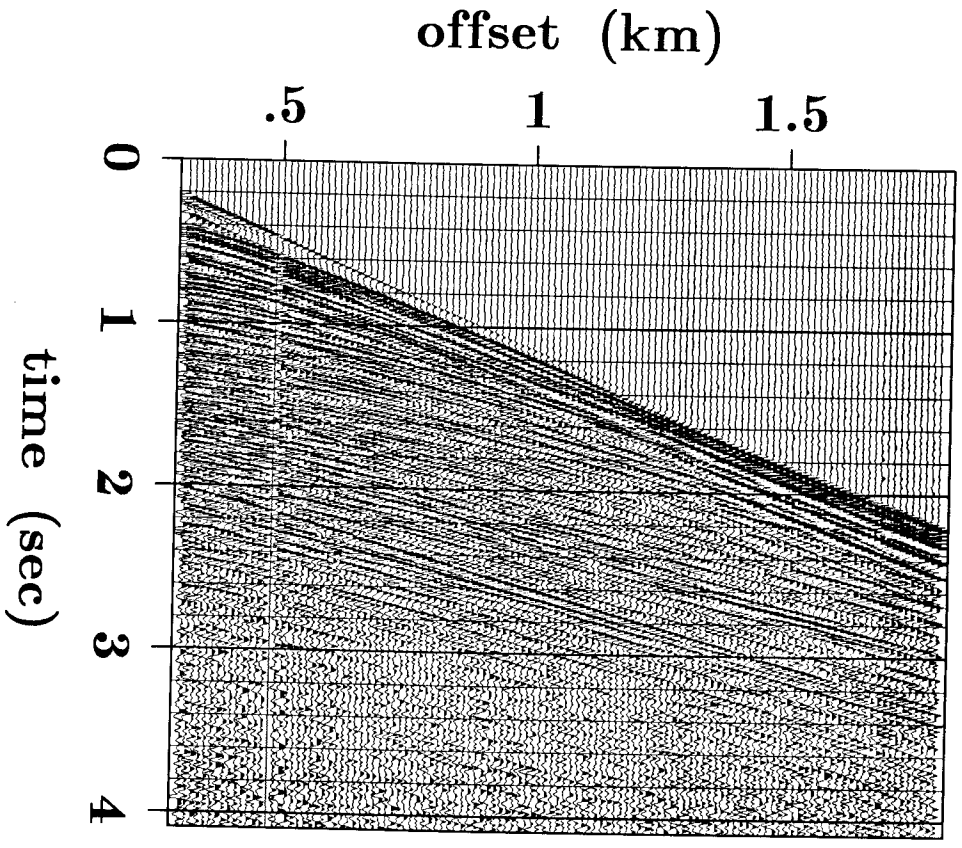


FIG. 14. A typical marine profile to be used in prestack migration velocity analysis.

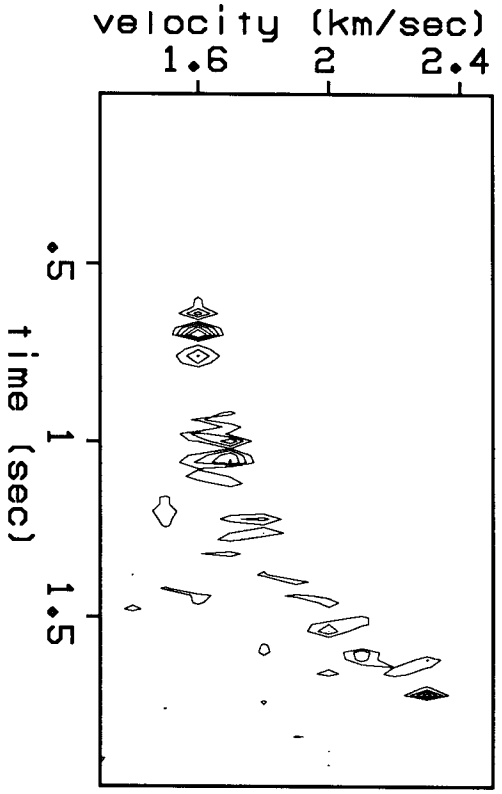


FIG. 15. Migrated and stacked CRG from the data in Figure 14 as a function of migration velocity.

the limitations of the technique. Now I will test the method on a marine data kindly provided by Chevron. The test is not complete and further work is needed in this area. The final comparison should be made between the velocity and the stack obtained from prestack migration and those obtained from conventional methods. This will be possible when determining the residual velocity and updating the velocity model is automated.

The profiles have 240 receivers each with receiver spacing of 12.5 m. The original shot spacing was 25 m. Since only every other profile was used, the shot spacing increased to 50 m. Figure 14 shows a typical profile. To test the search method, the profiles were migrated and stacked with 13 constant velocities. The result is shown in Figure 15 which looks similar to a conventional velocity spectrum. The peaks correspond to the rms velocity at the corresponding travel time. Qualitative comparison with some conventional velocity analysis for this data set shows similarity between the two.

To determine the residual velocity from one pass of migration (the iterative method), we need to look at a CRG migrated with a guessed velocity (say a constant velocity, or

velocity obtained from conventional velocity analysis). No result is shown here since work is still in progress.

CONCLUSIONS

We have seen that velocity can be obtained solely from pre-stack migration results. Such a velocity analysis method has less limitations than the conventional method. The medium velocity is obtained either by migrating with a guessed velocity and the residual velocity is determined, or by migrating and stacking with a suite of velocities and a velocity function is extracted from those velocities. Work is still needed to automate the process, particularly updating the model after determining the residual velocity. Results from synthetic examples and field data show that the method gives results that are at least as good as the conventional method. It is expected that the method works better in areas of complex structure.

It is well known that doing migration is more expensive than doing NMO. The cost is however decreasing fairly quickly, as computer speed and parallelism increases. At present, the method is too costly to be used in routine processing.

ACKNOWLEDGMENTS

I thank Francis Muir for many interesting discussions and useful suggestions.

REFERENCES

Al-Yahya, K., and Muir, F., 1984, Velocity analysis using prestack migration, SEP–41, p. 121-147.

Fowler, P., 1985, Migration velocity analysis by optimization: linear theory, SEP–44, p. 1-20.

Jacobs, A., 1982, The prestack migration of profiles, Ph.D. thesis, Stanford University.

Rothman, D., 1983, A short note on constant velocity migration, SEP–35, p. 127-131.

Rothman, D., 1985, Large near-surface anomalies and simulated annealing, Ph.D. thesis, Stanford University.

Thorson, J., 1980, Synthetic examples of prestack migration, SEP–24, p. 5-31.

APPENDIX

Profile migration in the frequency-wavenumber domain

Let $p(g,t,z=0)$ be the recorded data. First we Fourier transform the recorded data

$$P(k_g,\omega,z=0)=\iint p(g,t,z=0)e^{ik_g-i\omega t}dgdt$$

The first step in migration is to extrapolate the data to depth z,

$$P(k_g,\omega,z)=P(k_g,\omega,z=0)e^{-ik_z z} \quad .$$

We now time shift the extrapolated data. Before we do that, we have to inverse Fourier transform the data over the spatial axis

$$P'(g,\omega,z)=\int P(k_g,\omega,z)e^{-ik_g}dk_g$$

The time shift is done in the frequency domain

$$Q(g,\omega,z)=P'(g,\omega,z)e^{i\frac{\omega}{v}\sqrt{g^2+z^2}}$$

Now we Fourier transform the data over the time axis

$$q(g,t,z)=\int Q(g,\omega,z)e^{-i\omega t}d\omega$$

The migrated data (the reflectivity) is $q(g,t=0,z)$

$$\begin{aligned}R(g,z)=q(g,t=0,z)&=\int Q(g,\omega,z)d\omega\\&=\int P'(g,\omega,z)e^{i\frac{\omega}{v}\sqrt{g^2+z^2}}d\omega=\iint P(k_g,\omega,z)e^{-ik_g g}e^{i\frac{\omega}{v}\sqrt{g^2+z^2}}d\omega dk_g\\&=\iint P(k_g,\omega,z=0)e^{-ik_z z-ik_g g}e^{i\frac{\omega}{v}\sqrt{g^2+z^2}}d\omega dk_g\end{aligned}$$

Weird code, part 1

Below is a sample of some (deliberately) very weird programs which came over the net recently. All are in the C programming language, and supposedly compile and run on a Unix Vax. *Real* wizards, of course, will know what these programs do at a glance...

FROM THE NET

Here are some particularly worthwhile items to come across Usenet recently. Reprinted without anybody's permission, of course...

[] [] [] [] [] [] [] [] [] [] [] [] [] [] [] [] [] [] []

From: chongo@nsc.UUCP (Landon Noll)
Newsgroups: net.lang.c,net.unix-wizards
Subject: 2nd Annual Obfuscated Contest Winners
Organization: Rational Swamiconductor, Sanivale

*** OBFUSCATE THIS LINE WITH YOUR MESSAGE ***

The following programs were judged good (bad? wierd?) enough to win awards. This year, rather than trying to rank the programs in order of obfuscatedness, we gave a single award in each of 4 categories and a grand prize.

---

1. The most obscure program:
(submitted by Lennart Augustsson <seismo!mcvax!enea!chalmers!augustss>)

```
#define p struct c
#define q struct b
#define h a->a
#define i a->b
#define e i->c
#define o a=(*b->a)(b->b,b->c)
#define s return a;}q*
#define n (d,b)p*b;{q*a;p*c;
#define z(t)(t*)malloc(sizeof(t))
q{int a;p{q*(*a)();int b;p*c;}*b;};q*u n a=z(q);h=d;i=z(p);i->a=u;i->b=d+1;s
v n c=b;do o,b=i;while(!(h%d));i=c;i->a=v;i->b=d;e=b;s
w n o;c=i;i=b;i->a=w;e=z(p);e->a=v;e->b=h;e->c=c;s
t n for(;;)o,main(-h),b=i;}main(b){p*a;if(b>0)a=z(p),h=w,a->c=z(p),a->c->a=u,a->
c->b=2,t(0,a);putchar(b?main(b/2),-b%2+'0':10);}
```