Steep-dip $v(z)$ Imaging from an Ensemble of Stolt-like Migrations

by

Weston Mikulich

— Master's Thesis —

Center for Wave Phenomena
Department of Mathematical
and Computer Sciences
Colorado School of Mines
Golden, Colorado 80401
Phone (303) 273-3557

Steep-dip v(z) imaging from an ensemble of Stolt-like migrations

Weston Mikulich

ABSTRACT

Stolt migration is often performed with many different velocities, creating a cube (or *ensemble*) of migrated constant-velocity sections. With this ensemble, an interpreter can quickly, even interactively, carve out a single, variable-velocity time migration. Unfortunately, the variable-velocity section obtained in this way cannot image steep dips correctly.

Phase-shift (i.e. Gazdag) migration accurately images reflections from steep interfaces where velocity varies with depth. However, it is computationally slow compared to Stolt's f-k migration, its constant-velocity counterpart.

Similarities between the Gazdag and Stolt methods allow a mapping between the two to be derived. This new mapping, which replaces the constant-velocity Stolt mapping, defines a new *Stolt-like* migration. Stolt-like migration can then be performed to create an ensemble of migrated data, from which a *Gazdag-equivalent* migration can be carved. All dips can be imaged properly, even where velocity varies significantly with depth, while the advantages of working with the ensemble of migrated sections are preserved.

INTRODUCTION

Accurate imaging of seismic data requires accurate estimation of the velocity function used for migration. An effective method for estimating velocities is to migrate unstacked data a number of times, each time with a different constant velocity. This general approach is most efficiently done with the frequency-wavenumber domain method of Stolt (1978), which is both fast and accurate for constant-velocity migration. A cube of migrated data is produced, which will be referred to as the *Stolt ensemble*, having axes of distance, migrated time, and velocity. From this ensemble, a variable-velocity section can be carved. The carving process is fast, requiring merely an interpolation of existing data. Therefore, the interpreter can interactively

1

get migrated sections corresponding to many different trial velocity functions without having to re-migrate. This process is briefly described by Fowler (1984) and Li et al. (1991). Unfortunately, the section obtained by the carving process is generally not the same as the section properly migrated with a variable velocity function. The refraction of seismic waves that occurs due to velocity variations with depth is not accurately handled because the final image is obtained by interpolation between *constant-velocity* migrated results.

Phase-shift migration, pioneered by Gazdag (1978), has become a standard for post-stack migration accuracy where velocity is a function of depth only. In Gazdag's method, one extrapolates the wavefield in depth by performing phase shifts in the frequency-wavenumber domain. This method accurately images all dips up to 90 degrees and easily accommodates velocity variations with depth. Refraction (ray bending) that accompanies depth-varying velocity is treated exactly.

The ideal situation would be one where a migration algorithm properly handles ray bending yet retains all of the advantages of working with the Stolt ensemble. Li et al. (1991) describe a modification to the prestack Stolt method that enhances its accuracy in imaging steep reflections. Their approximate method for handling ray bending uses a *dip-dependent velocity* in the Stolt mapping process, with the approximation governed by the fourth-order Taylor-series approximation of moveout for depth-variable velocity.

Similarities between the Gazdag and Stolt migration methods suggests that ray bending can be accommodated without approximation. Here, an exact mapping is derived that enables an accurate, Gazdag-equivalent migration to be carved from the *ensemble of Stolt-like migrations*. The computational effort of this new method is on the same order as that of creating an ensemble of constant-velocity Stolt migrations.

While the Stolt-like ensemble proposed here is most appropriate when created from unstacked data, for simplicity, the considerations here are that the seismic data to be migrated are stacked, or, more precisely, zero-offset. The extension to prestack migration is outlined in Appendix A.

CONVENTIONAL STOLT ENSEMBLE MIGRATION

Migration velocities

The velocity model is one of the most important considerations when migrating seismic data. Often the velocity function is not well known, such as in frontier or structurally complex areas. In such areas it would be advantageous if the velocity function could remain somewhat arbitrary when doing migration. At least one way to achieve this goal is to perform many constant-velocity Stolt migrations at different velocities. One can later carve into the resulting sections to obtain a migration where the velocity varies spatially. New velocity functions can be tested without having to re-migrate any data. In this way, the "best" velocities for migration are found by simply migrating with numerous constant velocities.

Where velocity varies with depth, however, the method of migrating with constant velocities and carving a variable-velocity section can leave steep events undermigrated. For shallow dips, carving with the vertical-path, root-mean-square (rms) velocity, v_{rms}, is appropriate. However, the rms velocity actually varies with propagation direction, suggesting that simple interpolation of an ensemble of constant-velocity migrations is inadequate. Indeed, the "best" velocities for constant-velocity migration are dip-dependent.

To understand this dip-dependence of the migration velocities, consider a zero-offset seismic experiment over a point diffractor at a two-way vertical time τ in a medium where velocity increases with depth. The true traveltime t to this point can be expressed by the pair of parametric equations (1), where x represents horizontal distance from the diffractor, p is the ray parameter, and $v(\sigma)$ is the medium velocity as a function of vertical time. Dix (1955) demonstrated that an approximation to these equations yields the equation of a hyperbola, specified in equation (2). This approximation is valid for small distances x.

$$x = \int_0^\tau \frac{p\, v^2(\sigma)\, d\sigma}{4\sqrt{1 - \frac{p^2 v^2(\sigma)}{4}}} \qquad t = \int_0^\tau \frac{d\sigma}{\sqrt{1 - \frac{p^2 v^2(\sigma)}{4}}} \,. \tag{1}$$

$$t^2 = \tau^2 + \frac{4x^2}{v_{rms}^2} \,. \tag{2}$$

For typical variation of velocity with depth, the true traveltime differs from the Dix approximation hyperbola, much as shown in Figure 1. Diffracted energy arrives at times indicated by the actual traveltime curve. Migration should collapse all of the actual diffracted energy to a point, but if the rms velocity is used, amplitudes along the Dix hyperbola will be summed and placed at the apex, resulting in the wrong image.

The error in the Dix approximation becomes most severe for large lateral distances from the apex. This tendency follows from the fact that the vertical-path rms velocity function is appropriate only for the vertical raypath at $x = 0$. Other raypaths are not vertical and cannot correctly be accounted for by using the vertical rms velocity.

The exact traveltime curve is less steep and of earlier time than is the Dix hyperbola. In order to more closely approximate the flanks of the actual traveltime curve, the velocity in the Dix approximation would have to be increased. This suggests that in order to image steeply dipping events, a velocity that is higher than the rms velocity should be used. It also implies that the imaging velocity is somehow a function of dip.

In a medium where velocity varies with depth, ray bending occurs. To image steeply dipping events in their proper locations, this ray bending must be handled correctly, in accordance with equations (1). Gazdag migration properly handles the ray bending associated with depth-variable velocity, whereas Stolt migration does not. Since both approaches operate in the frequency-wavenumber domain, it is useful to look at the mathematical relationship between Gazdag and Stolt migration.
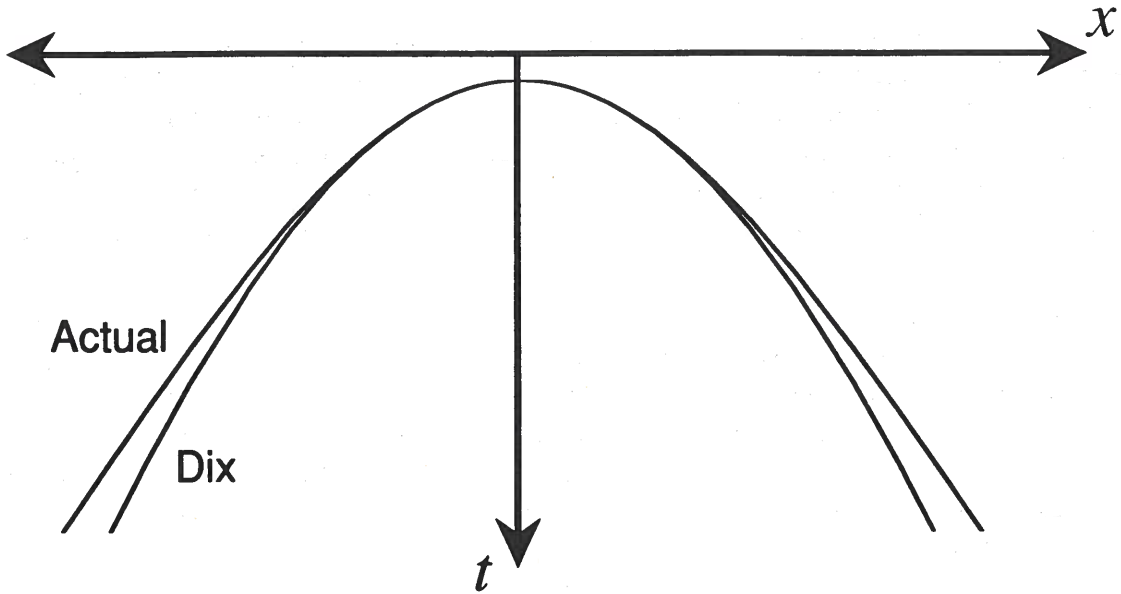
FIG. 1. The exact traveltime curve differs from that given by the Dix approximation. The larger moveout of the Dix approximation implies a velocity higher than v_{rms} is necessary to image steep dips.

Gazdag and Stolt migrations

Equation (3) demonstrates how to do Gazdag migration (Gazdag, 1978), and is derived in Appendix B. To obtain the migrated data q at a certain two-way vertical (migrated) time τ and horizontal wavenumber k, multiply the two-dimensional Fourier-transformed data $P(\omega, k)$ by a complex exponential, which contains an integral, and integrate over frequency ω.

$$q(\tau, k) = \frac{1}{2\pi} \int d\omega \, e^{-i\omega \int_0^\tau d\sigma \sqrt{1 - \frac{v^2(\sigma)k^2}{4\omega^2}}} \, P(\omega, k). \tag{3}$$

While Gazdag migration is accurate, it is computationally slow compared to Stolt migration. Unlike Gazdag migration, Stolt migration executes at fast Fourier transform (FFT) speeds. By making the assumption that velocity no longer varies with vertical time, as it does in equation (3), a Stolt migration equation can be obtained. Replacing the time-varying velocity $v(\sigma)$ by a constant velocity v_s, the migrated data can now be expressed as

$$q(\tau, k) = \frac{1}{2\pi} \int d\omega \, e^{-i\omega\tau \sqrt{1 - \frac{v_s^2 k^2}{4\omega^2}}} \, P(\omega, k). \tag{4}$$

If this integral were simply a Fourier transform, the computational speed of the FFT could be utilized in migrating the data. Stolt (1978) showed that by introducing a new frequency, ω_τ, this equation could be transformed to one that is Fourier-like. Equation (5) defines this change of variable.

$$\omega_\tau \equiv \omega \sqrt{1 - \frac{v_s^2 k^2}{4\omega^2}} = \text{sgn}(\omega) \sqrt{\omega^2 - \frac{v_s^2 k^2}{4}}. \tag{5}$$

The Jacobian of the transformation is

$$\frac{d\omega_\tau}{d\omega} = 2\,\delta(\omega)\,\sqrt{\omega^2 - \frac{v_s^2 k^2}{4}} + \frac{\omega\,\text{sgn}(\omega)}{\sqrt{\omega^2 - \frac{v_s^2 k^2}{4}}}.$$

The first term in the Jacobian has a nonzero value only at zero frequency. Zero frequency ($\omega = 0$) is absent from bandlimited recorded seismic data, so the first term may be disregarded. By making this change of variable, the migrated data become

$$q(\tau, k) = \frac{1}{2\pi} \int d\omega_\tau\, e^{-i\omega_\tau \tau}\, \frac{P\left(\omega_\tau \sqrt{1 + \frac{v_s^2 k^2}{\omega_\tau^2}}, k\right)}{\sqrt{1 + \frac{v_s^2 k^2}{\omega_\tau^2}}}. \tag{6}$$

This equation is simply a Fourier transform of a stretched and scaled version of $P(\omega, k)$, and may be implemented very efficiently via FFTs. Equation (6), which expresses Stolt migration, can be used to migrate seismic data at many different constant velocities v_s, in order to create an ensemble of constant-velocity Stolt migrations.

Sampling in velocity

The velocity axis of the Stolt ensemble must be sampled finely enough in order to prevent aliasing. Different velocities v_s map the same frequency ω to different migrated frequencies ω_τ, where equation (5) defines migrated frequency ω_τ as a function of velocity v_s. The change in the migrated frequency from one velocity to the next at one particular migrated time τ must be such that the change in phase is not more than half a cycle. Thus, to prevent aliasing, $|\Delta\omega_\tau \tau| < \pi$. $\Delta\omega_\tau$ can be obtained by differentiating equation (5) with respect to v_s, which yields

$$\frac{\partial\omega_\tau}{\partial v_s} = \omega\left(\frac{-v_s k^2}{4\omega^2}\right)\left[1 - \frac{v_s^2 k^2}{4\omega^2}\right]^{-1/2} \approx \frac{\Delta\omega_\tau}{\Delta v_s}.$$

By substituting in the anti-aliasing criterion for $\Delta\omega_\tau$, replacing ω with its frequency counterpart $2\pi f$, and solving for Δv_s, the following requirement is obtained to avoid velocity aliasing.

$$\Delta v_s < \frac{2\sqrt{1 - \frac{p^2 v_s^2(\tau)}{4}}}{f\,\tau\,p^2\,v_s(\tau)}, \tag{7}$$

where $p \equiv k/\omega$ is reflection slope. Inequality (7) must be satisfied to prevent aliasing of an event at migrated time τ, of frequency (before migration) f, and slope p. This criterion should be used to determine the proper sampling of v_s when computing an ensemble of constant-velocity Stolt migrations.

A CORRECTION TO CONSTANT-VELOCITY STOLT MIGRATION

Dip-dependent imaging velocity

Equations (3) and (4) are two different expressions for the migrated data as a function of migrated time τ and wavenumber k. The two methods would be equivalent

for a particular migrated time τ if, for that value of τ,

$$\int_0^\tau d\sigma \sqrt{1 - \frac{v^2(\sigma)k^2}{4\omega^2}} = \tau \sqrt{1 - \frac{v_s^2 k^2}{4\omega^2}}.$$

Solving for v_s^2 in the above equation and defining the slope $p \equiv k/\omega$ yields

$$v_s^2 = \frac{4}{p^2}\left[1 - \left(\frac{1}{\tau}\int_0^\tau d\sigma \sqrt{1 - \frac{v^2(\sigma)p^2}{4}}\right)^2\right]. \tag{8}$$

This equation, which is trivially satisfied for constant velocity, may be used in the presence of depth-variable velocity to obtain a Gazdag-equivalent migration by first computing an ensemble of constant-velocity migrations for different values of v_s. Then, for proper imaging of a slope p at time τ, this equation tells us what v_s to choose from the ensemble of constant-velocity Stolt migrations. This velocity will be referred to as the *imaging velocity*.

The presence of p in equation (8) implies that the imaging velocity is dip-dependent, and so it is best used for data in the frequency-wavenumber domain. Although the imaging velocity does depend on τ, this does not mean that a different constant-velocity Stolt migration is necessary for every time sample. Compared to the recorded seismic waves, the velocity v_s in equation (8) is a slowly varying function of τ. This allows a relatively coarse sampling of velocity without aliasing, as given by equation (7).

Approximations for small dips

To better understand the dip-dependent imaging velocity v_s, consider the two-term Taylor series expansion for the square-root in equation (8), which is a good approximation for small dips.

$$\sqrt{1 - \frac{v^2(\sigma)p^2}{4}} \approx 1 - \frac{v^2(\sigma)\,p^2}{8}$$

Substituting this approximation into equation (8),

$$v_s^2 \approx \frac{4}{p^2}\left\{1 - \left[\frac{1}{\tau}\int_0^\tau d\sigma \left(1 - \frac{1}{8}v^2(\sigma)p^2\right)\right]^2\right\}$$

$$\approx \frac{4}{p^2}\left\{1 - \left[\frac{1}{\tau}(\tau) - \frac{p^2}{4\tau}\int_0^\tau d\sigma\, v^2(\sigma)\right]\right\}$$

$$\approx \frac{1}{\tau}\int_0^\tau d\sigma\, v^2(\sigma)$$

$$\approx v_{rms}^2(\tau),$$

which is equivalent to the Dix approximation.

Often the rms velocity is the function used to carve a section from the ensemble of constant-velocity Stolt migrations, and this approximation shows that the rms velocity is appropriate for gentle dips. However, for steeper dips, equation (8) shows that the imaging velocity depends on dip. By making the following fourth-order approximation for the same square-root,

$$\sqrt{1 - \frac{v^2(\sigma)\,p^2}{4}} \approx 1 - \frac{v^2(\sigma)\,p^2}{8} + \frac{v^4(\sigma)\,p^4}{32},$$

and substituting this into equation (8), this dip-dependence is seen as

$$v_s^2 \approx v_{rms}^2(\tau) + \frac{p^2}{16}\left[\frac{1}{\tau}\int_0^\tau d\sigma\, v^4(\sigma) - v_{rms}^4(\tau)\right]. \tag{9}$$

The second term in this equation is never negative for any velocity function (see Appendix C). This dip-dependence agrees with the intuition derived from examining the diffraction traveltimes in Figure 1: to image steep dips, corresponding to large reflection slopes p, an imaging velocity somewhat higher than the rms velocity must be used. This fourth-order approximation is equivalent to the dip-dependent velocity used by Li et al. in their effort to improve the accuracy of Stolt migration.

Migrating with a dip-dependent velocity approximation

It has been established that the correct imaging velocity to use in carving a velocity surface from an ensemble of constant-velocity Stolt migrations is dip dependent and somewhat higher than the rms velocity, with the difference depending on reflector dip. But is the difference between these two velocities significant? The work of Li et al. demonstrates that for steep dips, the difference can be important. Figure 2, after Li et al., is a variable-velocity migration carved from an ensemble of constant-velocity Stolt migrations of synthetic data modeled from three horizontal beds and planar beds dipping at 30, 45, 60, and 75 degrees, where the medium velocity is linearly increasing with depth. The actual locations of the reflectors are shown by bold lines. The migration is acceptable for early times and shallow dips. However, the 75-degree reflector after migration is significantly dislocated from its true position.

Figure 3 is a migration of the same data as in Figure 2, but the fourth-order dip-dependent correction to the velocity of Li et al., equivalent to that given by equation (9), was used in computing the ensemble of Stolt migrations. While the more gently dipping events are imaged as before, the steeply dipping events are imaged closer to their true locations. These steeper events, nevertheless, are still undermigrated. This is because the fourth-order correction, while an improvement, is poorest for the steeper dips; it is a gentle- to moderate-dip approximation.

Analysis of the imaging velocity

Figure 4 shows the values of the correct imaging velocity, given by equation (8), where medium velocity increases linearly with depth, for three different dips; a horizontal reflector and reflectors dipping at 60 and 85 degrees. For zero dip, the imaging
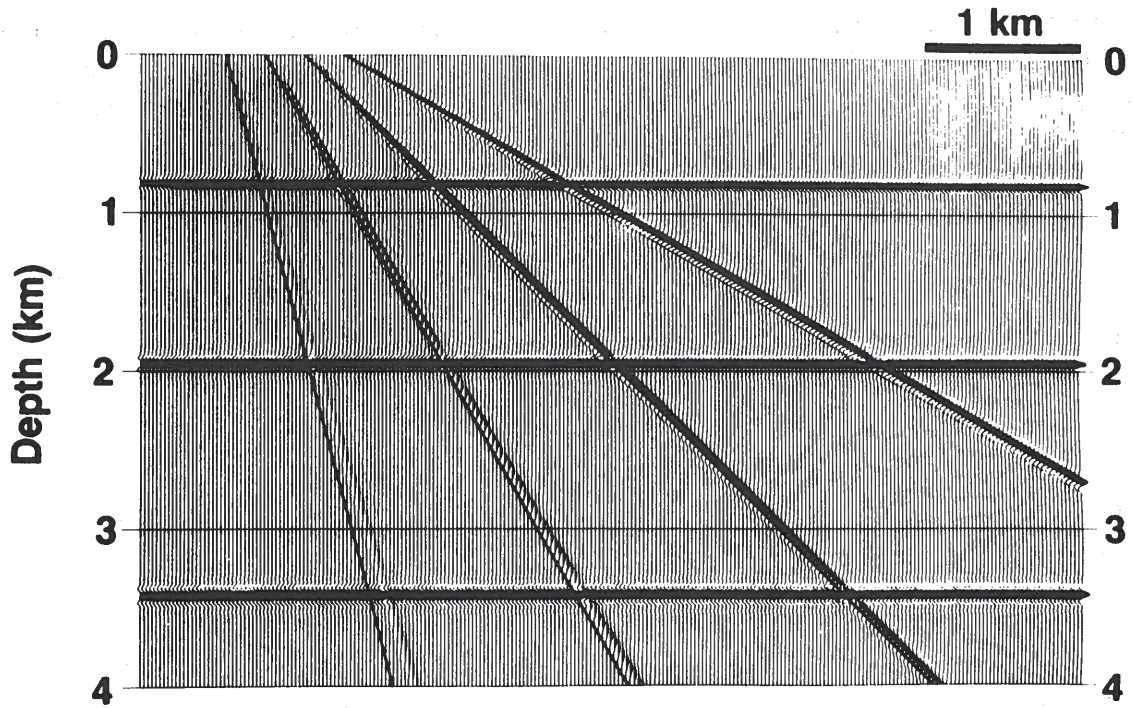
FIG. 2. Migration of synthetic data from seven beds, three horizontal and the other four dipping at 30, 45, 60, and 75 degrees. The rms velocity was used to carve the ensemble of constant-velocity Stolt migrations. (Li, et al., 1991)
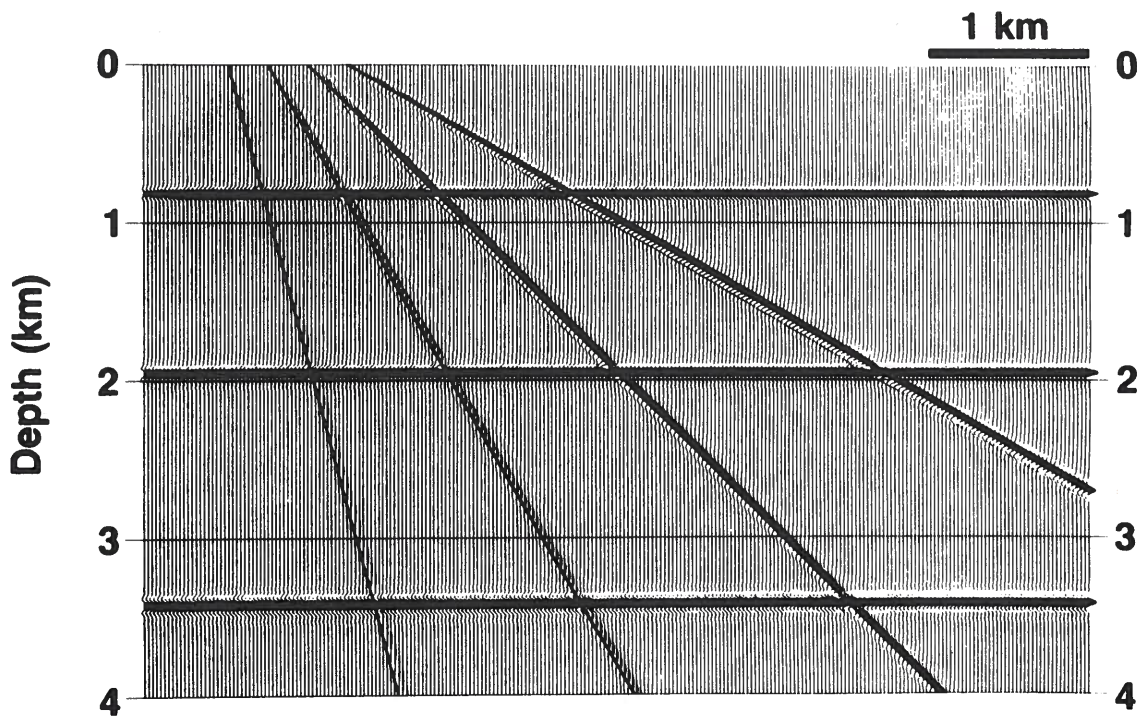

FIG. 3. Migration of synthetic data from seven beds, three horizontal and the other four dipping at 30, 45, 60, and 75 degrees. The fourth-order correction to the imaging velocity was used to create the ensemble of Stolt migrations. (Li, et al., 1991)

velocity is equivalent to the much more familiar rms velocity, as shown earlier. Note also that the value of the imaging velocity exceeds that of the rms velocity for steeper dips. Although this difference between the rms velocity and the imaging velocity appears to be small, it is what gives rise to the undermigration of the dipping events in Figure 2.
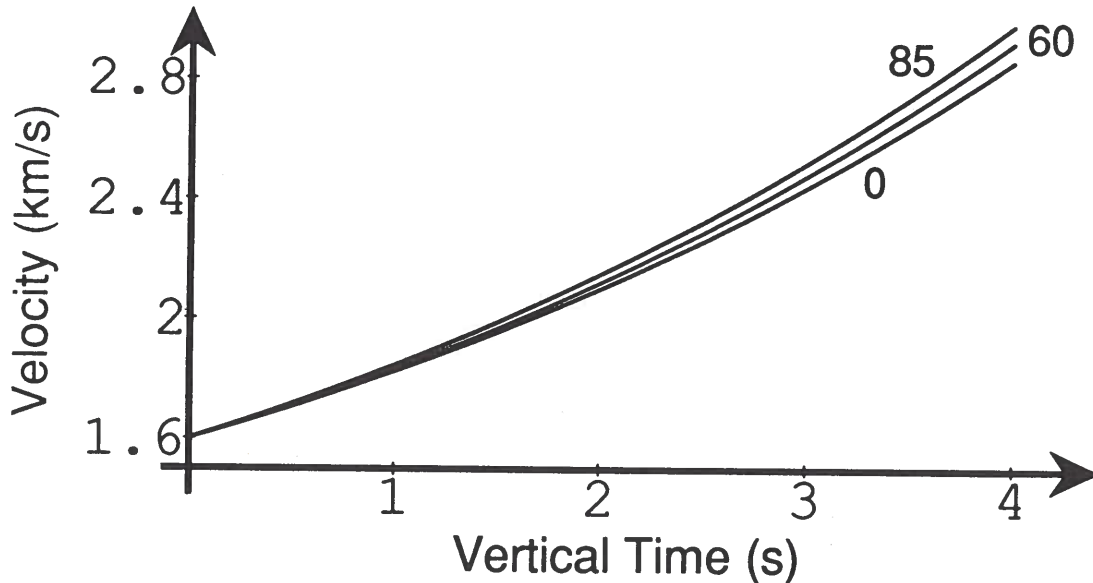

FIG. 4. Imaging velocity as a function of two-way vertical time τ for dips of 0, 60, and 85 degrees. The velocity function used was $v(z) = 1.6 + z/2$.

Figure 5 shows the percentage difference between the rms velocity and two other velocities – the correct imaging velocity v_s and the fourth-order approximation – for a dip of 85 degrees. As expected, the difference between the rms velocity and the imaging velocity increases with reflection slope. Note that the fourth-order correction accounts for less than 40% of the difference between the correct imaging velocity and the rms velocity. Yet, recall that for the less-steeply dipping events seen in Li, et al., the fourth-order correction was quite good.

Although the velocity differences appear to be small, the associated errors in reflector position are significant for steeply dipping events. For example, consider migrating a reflection from a reflector with a dip of 85 degrees at 3 s vertical time. After migration with the rms velocity, this reflector would appear misplaced horizontally (downdip) from its true position by 200 m, perhaps 16 traces, assuming a 12.5-m trace spacing. Figure 6 shows this lateral mispositioning of dipping events as a function of vertical time. This error is significant and substantiates the use of the dip-dependent imaging velocity.

Obtaining Figure 6 is no trivial task. The mispositioning at one particular vertical time τ is *not* merely the lateral separation between the true traveltime curve and the Dix hyperbola, as plotted in Figure 1. The mispositioning must be obtained by considering the reflection from a single dipping reflector. The Gazdag method would migrate energy from this reflection along the true traveltime curve (given by
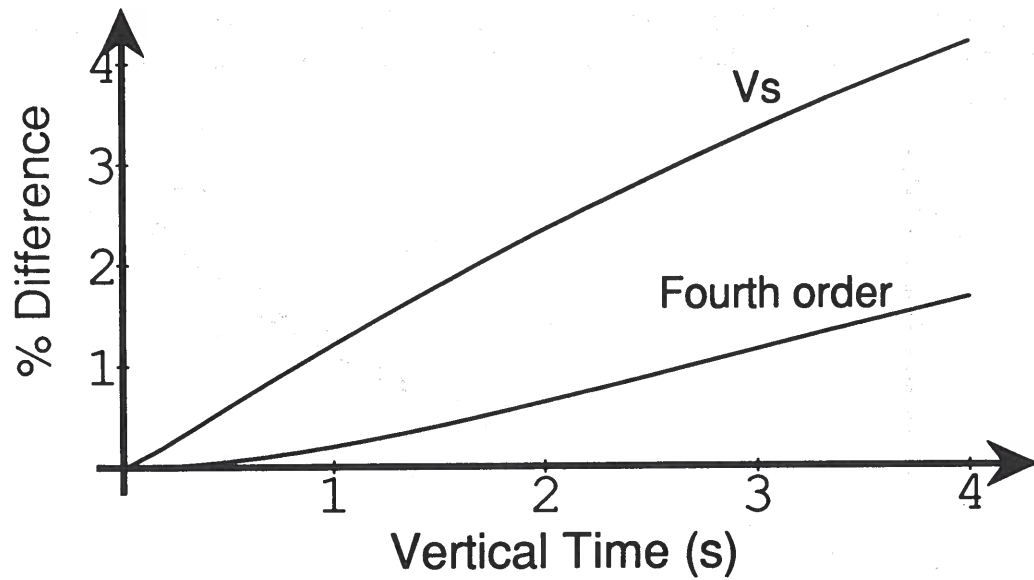
FIG. 5. Percentage difference between the rms velocity and two other velocities: the correct imaging velocity v_s and the fourth-order approximation. Here, the event to be imaged is a flat reflector with a dip of 85 degrees.
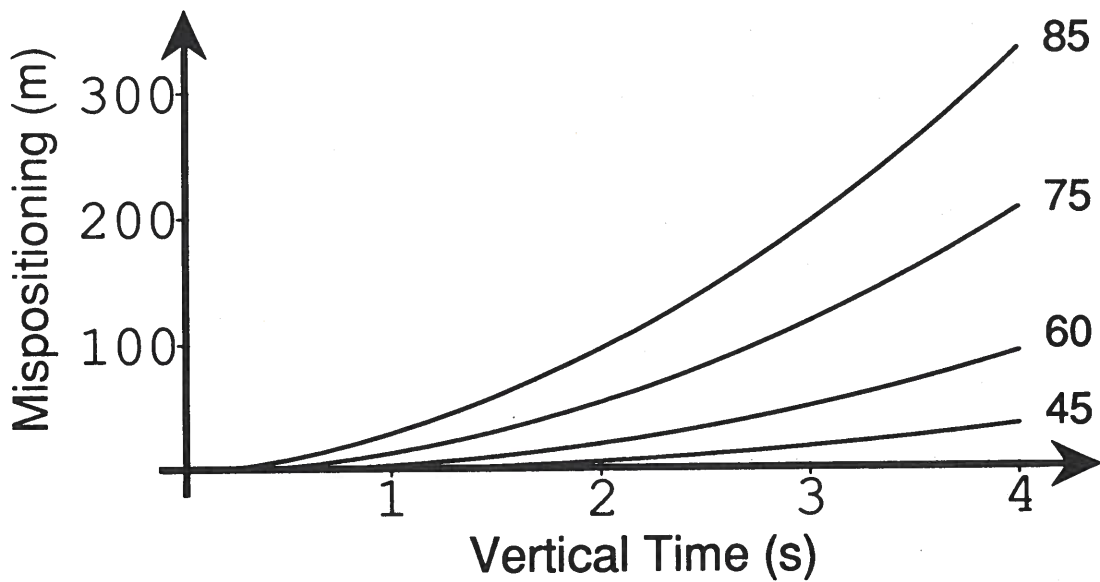

FIG. 6. Lateral mispositioning of events migrated with the rms velocity. Dips of 45, 60, 75, and 85 degrees are represented.

equations (1)) to the correct migrated time (τ) and lateral position. Stolt migration would move energy from a different point on the reflection via the Dix hyperbola, to the same apex time τ, but to a different lateral position. The difference at that τ between the apexes of these two curves is the mispositioning error. The difficulty in determining the lateral mispositioning arises from the need to solve for the point where the Dix hyperbola is tangent to the reflection.

THEORY OF STOLT-LIKE ENSEMBLE MIGRATION

A new Stolt-like migration

So far, it has been shown that in order to image steep dips properly where velocity varies with depth, the imaging velocity in the Stolt process must vary with dip. Equation (8) for the imaging velocity can be incorporated within the Stolt mapping of equation (6), which yields a *new* migration method. This new method is similar to the Stolt method, yet provides the accuracy of Gazdag migration. Recall that Gazdag migration correctly handles ray bending where velocity varies with vertical time, and can be written as

$$q(\tau, k) = \frac{1}{2\pi} \int d\omega \, e^{-i\omega \int_0^\tau d\sigma \sqrt{1 - \frac{v^2(\sigma)k^2}{4\omega^2}}} P(\omega, k). \tag{10}$$

Stolt's approach suggests the following change of variable in order to make the Gazdag equation look like a Fourier transform.

$$\omega_\tau \tau = \omega \int_0^\tau d\sigma \sqrt{1 - \frac{v^2(\sigma)k^2}{4\omega^2}}.$$

This defines the new frequency ω_τ to be

$$\omega_\tau \equiv \frac{\omega}{\tau} \int_0^\tau d\sigma \sqrt{1 - \frac{v^2(\sigma)k^2}{4\omega^2}}. \tag{11}$$

By making the above change of variable, and suppressing the wavenumber dependence of both the unmigrated and migrated data (although this equation is valid for all k), the migrated data become

$$q(\tau) = \frac{1}{2\pi} \int d\omega_\tau \, e^{-i\omega_\tau \tau} \tilde{Q}(\omega_\tau, \tau), \tag{12}$$

where

$$\tilde{Q}(\omega_\tau, \tau) = \left| \frac{\partial \omega}{\partial \omega_\tau} \right| P\left(\omega(\omega_\tau, \tau) \right),$$

and where $\omega(\omega_\tau, \tau)$ is defined implicitly by equation (11). Equation (12) would be a simple Fourier transform if it were not for the dependence of $\tilde{Q}$ on τ. Replace τ with a different variable s to avoid confusion between this τ and the Fourier dual variable of ω_τ, which also has the name τ. This yields

$$\tilde{Q}(\omega_\tau, s) = \left| \frac{\partial \omega}{\partial \omega_\tau} \right| P\left(\omega(\omega_\tau, s) \right). \tag{13}$$

For any value of s, $\tilde{Q}$ has an inverse Fourier transform $\tilde{q}$, given by

$$\tilde{q}(\tau, s) = \frac{1}{2\pi} \int d\omega_\tau \, e^{-i\omega_\tau \tau} \, \tilde{Q}(\omega_\tau, s),\tag{14}$$

where

$$\omega_\tau \equiv \frac{\omega}{s} \int_0^s d\sigma \sqrt{1 - \frac{v^2(\sigma)k^2}{4\omega^2}}.\tag{15}$$

In replacing τ by s in the argument of $\tilde{Q}$, integral expression (12) for the desired migrated result $q(\tau)$ has been replaced by a Fourier transform, equation (14), for the new function $\tilde{q}(\tau, s)$. Note, however, that when $s = \tau$, $\tilde{q}(\tau, s = \tau) = q(\tau)$. Thus, one way of computing $q(\tau)$ is to perform a number of inverse Fourier transforms (using FFTs) for many different constant values of the parameter s and subsequently select (actually, interpolate) from the volume of data generated the surface corresponding to $s = \tau$. This seemingly indirect way of solving equation (12) is analogous to the constant-velocity method of Stolt migration, in which the FFT is used to migrate data with many values of constant velocity v_s and a final migration result is obtained by interpolating from the volume of data along a surface defined by $v_s = v_{rms}$.

To use the relationship between ω and ω_τ in a migration algorithm it would be inefficient to evaluate the integral in equation (15) for every ω that was needed. Not only that, but equation (15) must be solved for ω for use in equation (13), which is not possible analytically. Solving equation (15) for ω_τ/ω,

$$\frac{\omega_\tau}{\omega} = \frac{1}{s} \int_0^s d\sigma \sqrt{1 - \frac{v^2(\sigma)k^2}{4\omega^2}}.\tag{16}$$

Values of ω_τ/ω, a function of s and $p \equiv k/\omega$, are calculated and placed in a table. A migration code that implements this new mapping would build up a two-dimensional array of values for ω_τ/ω, to be consulted as necessary during the migration. Linear interpolation can be used within this table to obtain the necessary values of $\omega(\omega_\tau, s)$, because values of equation (16) vary smoothly in the s and p directions. However, as $p \to 2/v(s)$, this function becomes infinitely steep, so care must be exercised in sampling this function near the evanescent edge, defined by $p = 2/v(s)$.

Sampling in the s dimension

Equation (15) is slowly varying in s, and a constant-s migration need not be computed for every time sample in the section to be migrated. With the s dimension sampled relatively coarsely, sinc interpolation at constant τ can be used effectively between these s planes. However, aliasing in the s dimension must be avoided. If the sampling in s is too coarse, the steeper events cannot be properly carved from the ensemble because they are aliased from one s-section to the neighboring s-section. Figure 7 demonstrates the result when the sampling interval in the s dimension is too large. The steeper events no longer appear as one coherent reflection, but rather as many disjointed events.
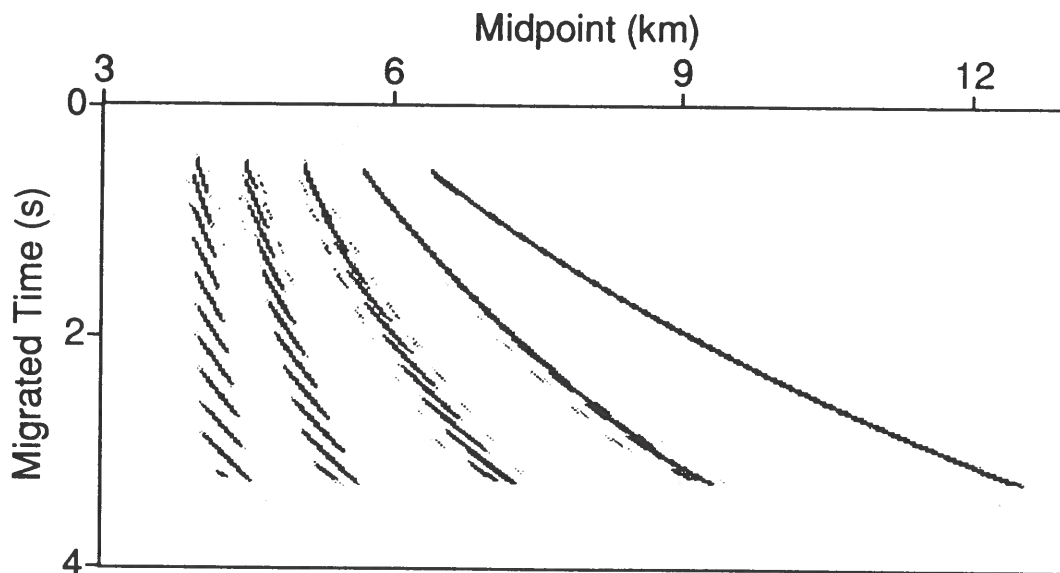
FIG. 7. Steeper events are aliased if the sampling in the s dimension is not fine enough.

In order to prevent aliasing in the s dimension, the change in the migrated frequency from one s to the next at one particular migrated time τ must be such that the change in phase is not more than half a cycle. In mathematical terms, $|\Delta\omega_\tau s| < \pi$. Migrated frequency is defined in equation (15) as

$$\omega_\tau = \frac{\omega}{s} \int_0^s d\sigma \sqrt{1 - \frac{v^2(\sigma)p^2}{4}}.$$

The change in migrated frequency with respect to s is given by the partial derivitive

$$\frac{\partial\omega_\tau}{\partial s} = \frac{\omega}{s}\left[\sqrt{1 - \frac{v^2(s)p^2}{4}} - \frac{1}{s}\int_0^s d\sigma \sqrt{1 - \frac{v^2(\sigma)p^2}{4}}\right] \approx \frac{\Delta\omega_\tau}{\Delta s}.$$

By substituting in the aliasing criterion, replacing ω with its frequency counterpart $2\pi f$, and solving for Δs,

$$\Delta s < \frac{1}{2f\left[\sqrt{1 - \frac{v^2(s)p^2}{4}} - \frac{1}{s}\int_0^s d\sigma \sqrt{1 - \frac{v^2(\sigma)p^2}{4}}\right]}. \qquad (17)$$

Inequality (17) must be satisfied to prevent aliasing of an event at migrated time s, of frequency (before migration) f, and slope p.

To image events dipping at 60 degrees and at 2 s of vertical time, equation (7) dictates that it would take 32 constant-velocity migrations, while equation (17) says that 44 constant-s migrations are necessary to prevent aliasing, where $v(z) = 1.6 + z/2$. In this case there is a 38% increase in the number of migrations necessary when performing Stolt-like ensemble migration. The sampling intervals in v_s and s depend

on v_s and s themselves, which makes it possible to optimize both Stolt ensemble migration and Stolt-like ensemble migration by allowing the sampling interval in v and s to vary so that fewer migrations actually need to be computed. This variable sampling has not been implemented here, though. Figure 8 illustrates the number of migrations necessary at $\tau = 2$ s as a function of the dip to be imaged for Stolt ensemble migration with a constant sampling in velocity and for Stolt-like ensemble migration with a constant sampling in s. The difference between these two curves is greater for the steeper dips.
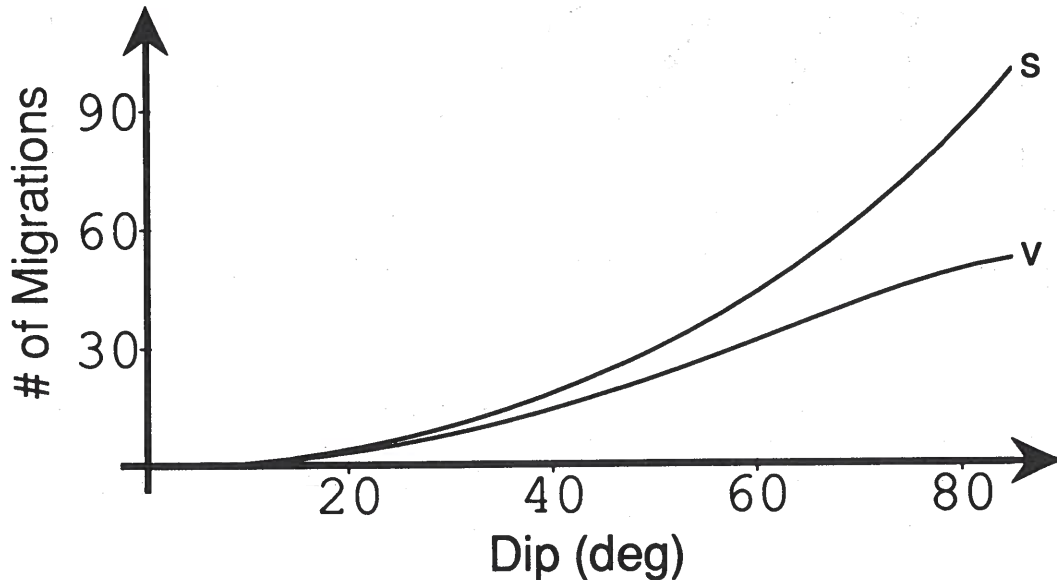

FIG. 8. Number of migrations necessary to prevent aliasing as a function of dip for a vertical two-way time of 2 s. For a constant sampling interval, more Stolt-like migrations (s) than constant-velocity Stolt migrations (v) are necessary.

U ensembles

The location of the data of interest in the Stolt-like ensemble should be in the vicinity of $s = \tau$, if the velocity function used for migration is close to the true propagation velocity of the earth. It is therefore unnecessary to provide as output the entire ensemble of migrated data for all values of s. By defining a new variable u equal to s/τ, and calculating the ensemble for a range of u's in the vicinity of $u = 1$, the predominant data of interest are obtained. This procedure is pictured in Figure 9.

The Gazdag-equivalent migration for the velocity function used for Stolt-like ensemble migration lies along $u = 1$ when the correct velocity is assumed in the mapping given by equation (16), but allowing u to vary with τ yields a migration from a different velocity function. Choosing $u < 1$ is equivalent to using a slower velocity at that τ, and using $u > 1$ is the same as migrating with a higher velocity at that τ.
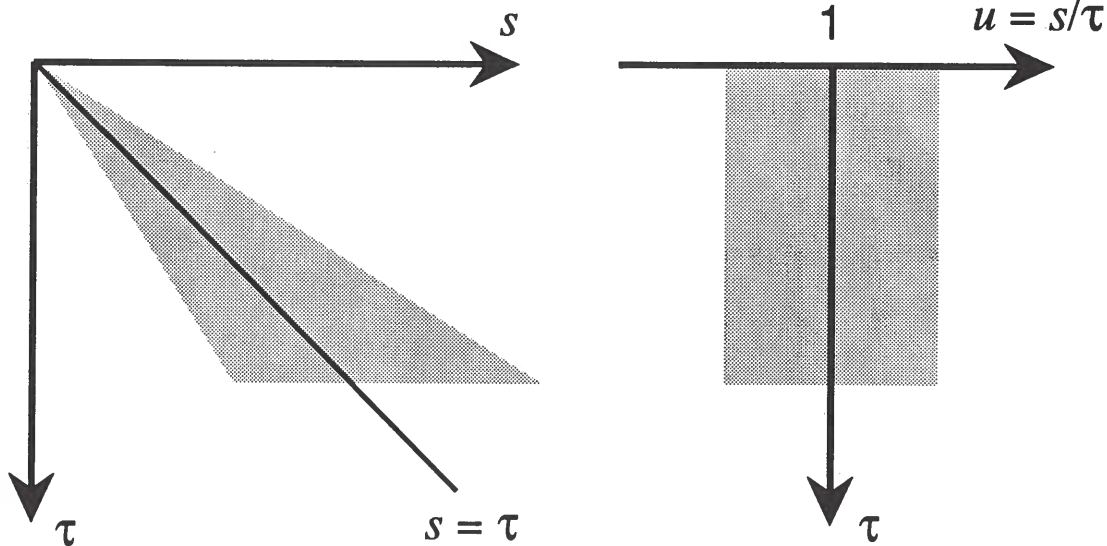
FIG. 9. Providing all of the s sections is not necessary; mapping the data from s to $u = s/\tau$ most efficiently samples the data of interest. Data are mapped from the shaded area in the s domain into the shaded area of the u domain.

Creating the ensemble with the wrong velocity

If the incorrect velocity function is used in computing the Stolt-like ensemble, one may still be able to carve an acceptable migration from the ensemble. However, the best image may no longer lie along the plane of $s = \tau$, or $u = 1$. In general, u varies with vertical time τ and slope p. But for some special cases, all dips can be properly imaged at all times, even though the wrong velocity was used in the frequency mapping which accomplishes migration.

Consider creating the Stolt-like ensemble with the incorrect velocity, $\tilde{v}(\sigma)$. The correct migration would be obtained by using the correct velocity function in equation (16), but the table would now be computed from the equation

$$\frac{\tilde{\omega}_\tau}{\omega} = \frac{1}{s} \int_0^s d\sigma \sqrt{1 - \frac{\tilde{v}^2(\sigma)p^2}{4}}. \tag{18}$$

To find what s to use in order to properly image slope p, set $\tilde{\omega}_\tau/\omega = \omega_\tau/\omega$. This yields a transcendental equation for s, seen as

$$\frac{1}{\tau} \int_0^\tau d\sigma \sqrt{1 - \frac{v^2(\sigma)p^2}{4}} = \frac{1}{s} \int_0^s d\sigma \sqrt{1 - \frac{\tilde{v}^2(\sigma)p^2}{4}}.$$

Substituting $u = s/\tau$ shows where the correct image lies in the u domain at migrated time τ for a particular slope p:

$$\frac{1}{\tau} \int_0^\tau d\sigma \sqrt{1 - \frac{v^2(\sigma)p^2}{4}} = \frac{1}{\tau u} \int_0^{\tau u} d\sigma \sqrt{1 - \frac{\tilde{v}^2(\sigma)p^2}{4}}.$$

15

By making the change of variable $\sigma' \equiv u\,\sigma$,

$$\frac{1}{\tau}\int_0^\tau d\sigma \sqrt{1 - \frac{v^2(\sigma)p^2}{4}} = \frac{1}{\tau}\int_0^\tau d\sigma' \sqrt{1 - \frac{\tilde{v}^2(u\,\sigma')p^2}{4}}.$$

This integral expression will be satisfied if

$$v(\tau) = \tilde{v}\left(u\,\tau\right). \qquad (19)$$

The variable substitution above is valid only if u is constant and not a function of τ. Thus, if the best migration happens to occur in the ensemble at a single constant value of u, equation (19) gives the true velocity function. This value of u does not vary with slope p, and therefore images all dips properly even when the wrong velocity is used in computing the Stolt-like ensemble. A Gazdag-equivalent migration can still be obtained which images all dips correctly.

Unfortunately, the best migration will occur for a constant value of u (time invariant) *only* if the velocity function is of a special form. In particular, if $v(\tau) = g(b\,\tau)$, and the wrong velocity $\tilde{v}(\tau) = g(\tilde{b}\,\tau)$ was used for migration, then $u = b/\tilde{b}$. But equation (19) could be used in a much more general way, even if u was found to vary with τ. Although not exact, this equation could be used to update the estimate of the velocity function.

For gentle dips, this special functional form of the velocity is not required in order to update the velocity function. Consider the small p approximation for both $\tilde{\omega}_\tau/\omega$ and ω_τ/ω. By setting these approximations equal to each other,

$$1 - \frac{p^2}{2\tau}\int_0^\tau d\sigma\, v^2(\sigma) = 1 - \frac{p^2}{2s}\int_0^s d\sigma\, \tilde{v}^2(\sigma).$$

Rewriting, $v_{rms}(\tau) = \tilde{v}_{rms}(s) = \tilde{v}_{rms}\left(\tau\,u(\tau)\right)$. This would indicate how changes in velocity relate to changes in u. For example, suppose Stolt-like ensemble migration is performed with the wrong velocity function corresponding to $\tilde{v}_{rms}(\tau)$. After examining the ensemble, a desirable migration is found by carving along a particular $u(\tau)$. An improved estimate of v_{rms} is then $v_{rms}(\tau) = \tilde{v}_{rms}\left(\tau\,u(\tau)\right)$. In this way, the imaging of gently dipping reflectors can be used to refine estimates of the velocity function.

IMAGING WITH A STOLT-LIKE ENSEMBLE

The process of Stolt-like ensemble migration proposed here is similar to performing many constant-velocity Stolt migrations. Figure 10 provides a summary of how to do migration with the Stolt-like mapping. The algorithm seen there was used as the basis for the computer program listed in Appendix D.

This ensemble migration has been tested on synthetic data from an earth model consisting of five planar reflectors with dips of 30, 45, 60, 75, and 85 degrees. The four

Supply velocity function $v(\tau)$
Build ω_τ/ω table for all s and $p = k/\omega$
Load data (synthetic or recorded seismic data)
FFT x to k
for all k {
    for all s in table {
        for all ω_τ {
            build $\omega_\tau(\omega)$ array
        }
        for all ω {
            Invert to get $\omega(\omega_\tau)$
            Interpolate, shift, etc. to get migrated data
        }
    }
}
for all u {
    FFT k to x
}
Write 3-d ensemble of migrated data

FIG. 10. Algorithm for creating a Stolt-like ensemble of constant-u migrations.

smaller dips correspond to the four dipping reflectors used by Li et al. as in Figures 2 and 3.

Gazdag migration is the accuracy standard with which to compare the accuracy of alternative migrations when velocity varies with depth. Therefore, in order to check the accuracy of the Stolt-like ensemble migrations, a Gazdag migration was also performed.

Imaging five dipping planes

Kirchhoff modeling was used to create synthetic zero-offset data corresponding to five dipping planes, of 30, 45, 60, 75, and 85 degrees. The velocity function used in modeling was $v(z) = 1.6 + z/2$. This synthetic is illustrated in Figure 11. Migrating the synthetic data of the five reflectors by the Gazdag method produces the image in Figure 12, with all reflectors correctly positioned. This will be used as the desired output in measuring the accuracy of Stolt-like ensemble migration.
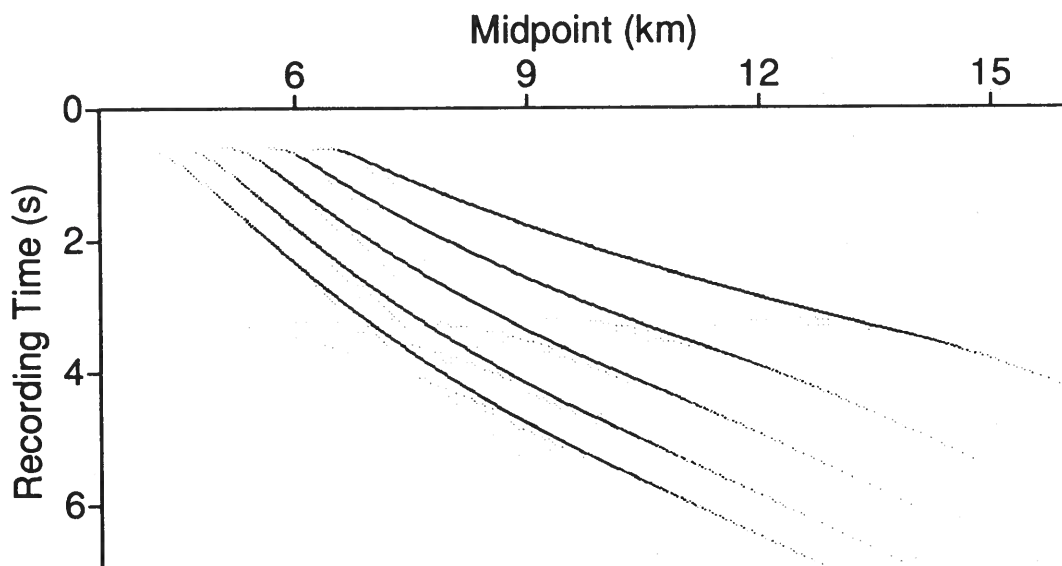

FIG. 11. Synthetic zero-offset data of five dipping planes, of 30, 45, 60, 75, and 85 degrees, obtained from Kirchhoff modeling.

The Stolt ensemble migration of the synthetic above is provided in Figure 13. This is the section that would be obtained if one did constant-velocity Stolt migration to build an ensemble of migrated data and later carved this ensemble using the rms velocities. Note the undermigration of the steeper dips, particularly evident at late time. Figure 14 shows a closeup region of the data migrated with both the Gazdag and Stolt methods. The Stolt migration exhibits significant error in the position of the steeper events. This error is consistent with that plotted in Figure 6.

By performing Stolt-like ensemble migration on the synthetic data using the known velocity in the mapping, Figure 15 is obtained. This is the section carved from the ensemble that is equivalent to $u = 1$, or using the theoretically correct velocity
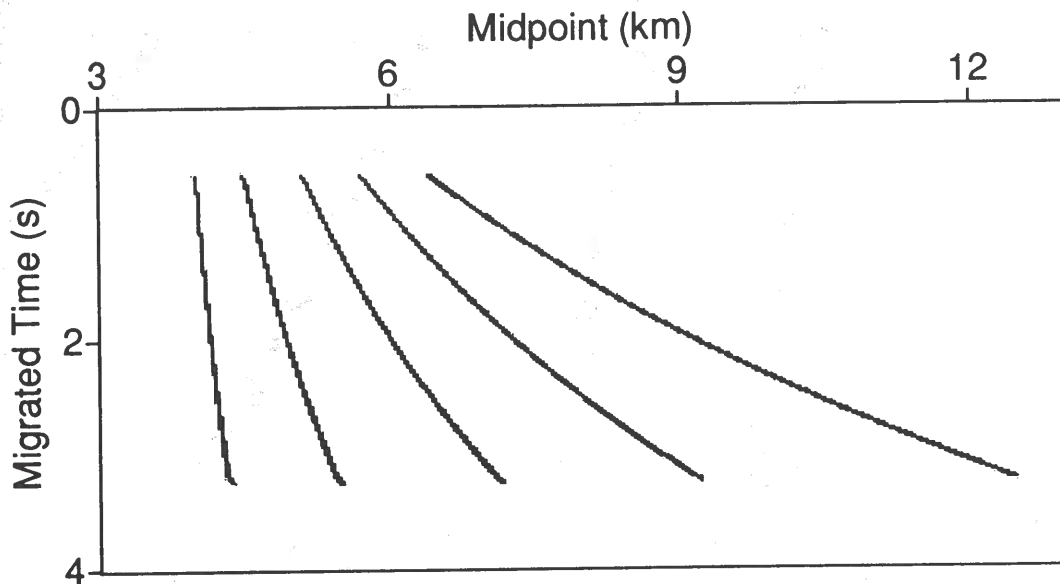
FIG. 12. Gazdag migration of the synthetic data of the five dipping reflectors, to be compared with the model.
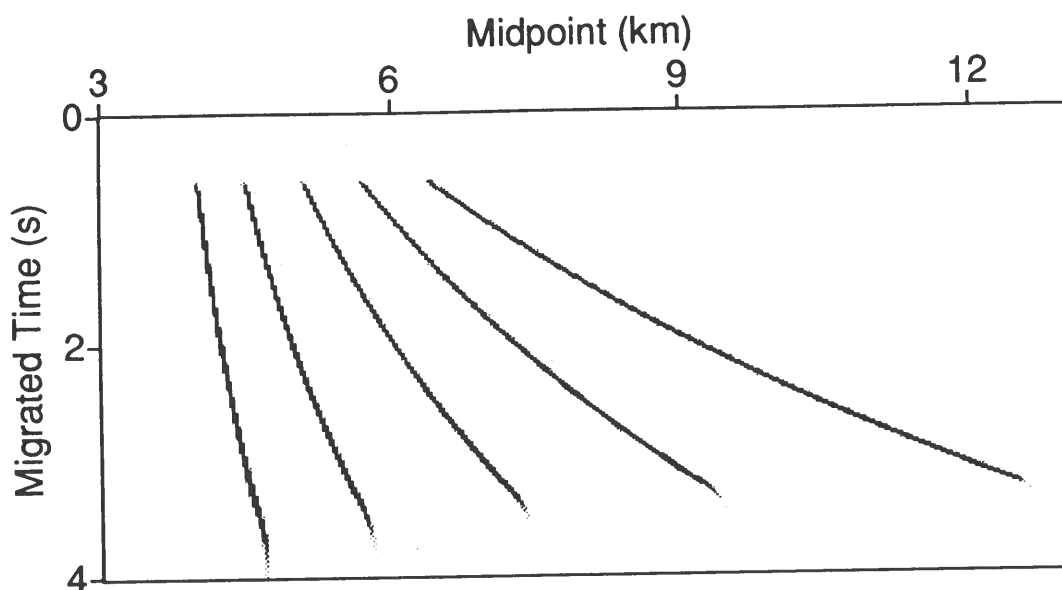

FIG. 13. Stolt ensemble migration of the synthetic Kirchhoff data. Undermigration increases with the dip of the reflector.
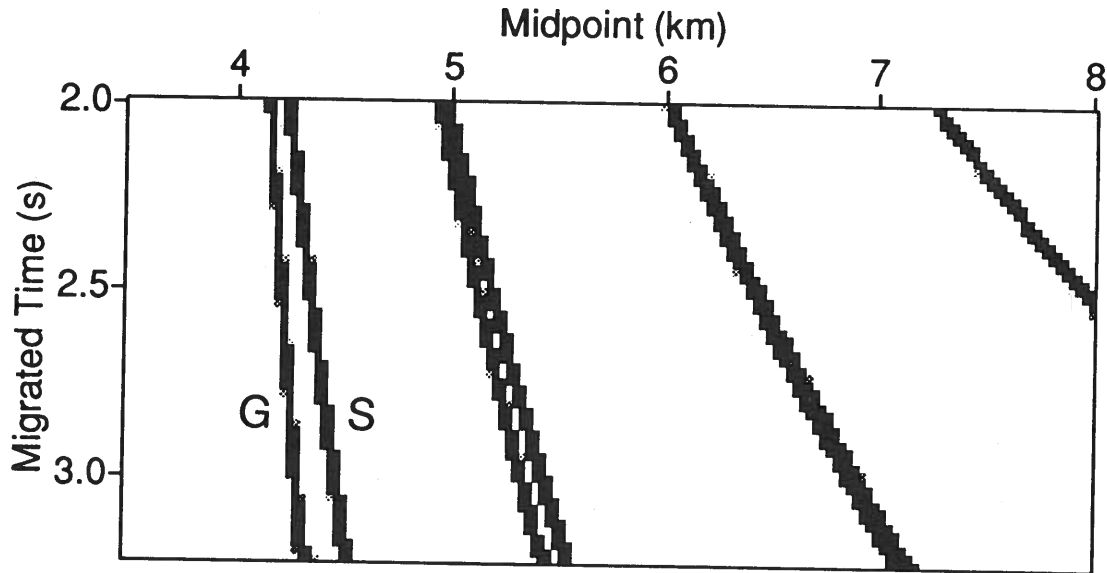
FIG. 14. Closeup of the migrations from the Gazdag and Stolt methods. Dips of 45, 60, 75, and 85 degrees can be seen. The mispositioning error in Stolt migration is most severe for the steeper dips and later times.

function with which the frequency mapping was done. Reflector positions for the Stolt-like ensemble method are the same as those from the Gazdag migration.

The output of Stolt-like ensemble migration is a function of τ, x, and u. By slicing into this ensemble at one particular value of x, one can see that the events for $u < 1$ are undermigrated, and the events where $u > 1$ are overmigrated. Figure 16 is a slice through the ensemble at $x = 5.2$ km, along with the Gazdag output at that position. At first this picture would seem to be in error; the events arrive at earlier times for the undermigrated case. But in fact when the events are undermigrated, they are somewhat left (downdip) of the true position, making them appear earlier at one particular x location.

To look at the lateral shift in x as a function of u, consider the (migrated) time slice illustrated in Figure 17. Again the correct Gazdag migration is provided for comparison. The undermigrated events appear at a greater distance downdip, or more toward the positive x direction. Also note the steeper events are more sensitive to velocity, as indicated by their larger lateral shift as a function of u.

CONCLUSION

Thesis summary

It is possible to carve a variable-velocity migration from an ensemble of constant-velocity Stolt migrations, but only small dips will be imaged properly. Similarities between Gazdag's phase-shift method and Stolt's method suggest a mapping to image all dips up to 90 degrees, where velocity varies with depth, by using an ensemble of
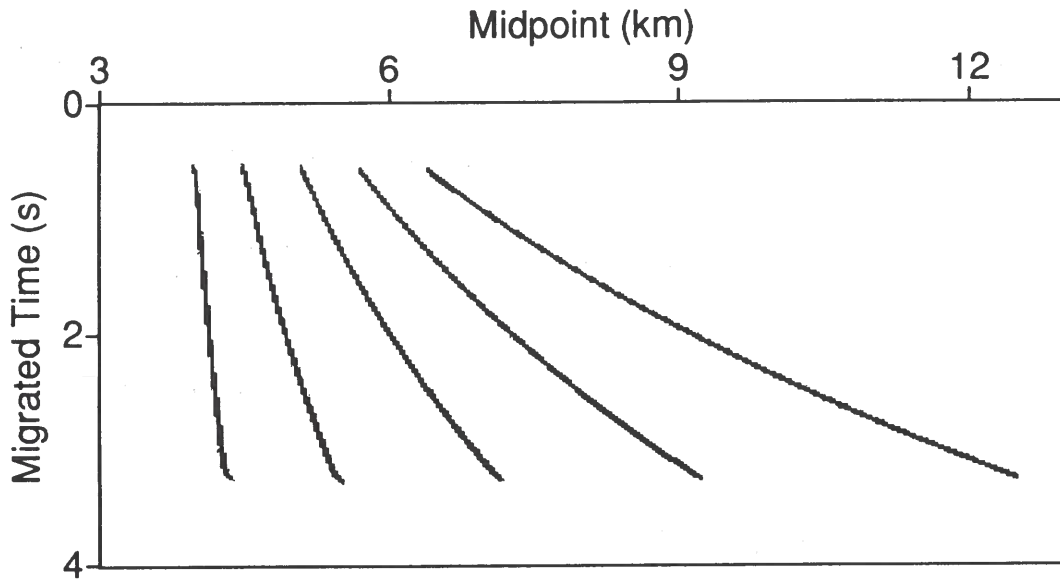
FIG. 15. Stolt-like ensemble migration of the synthetic data of the five dipping reflectors.
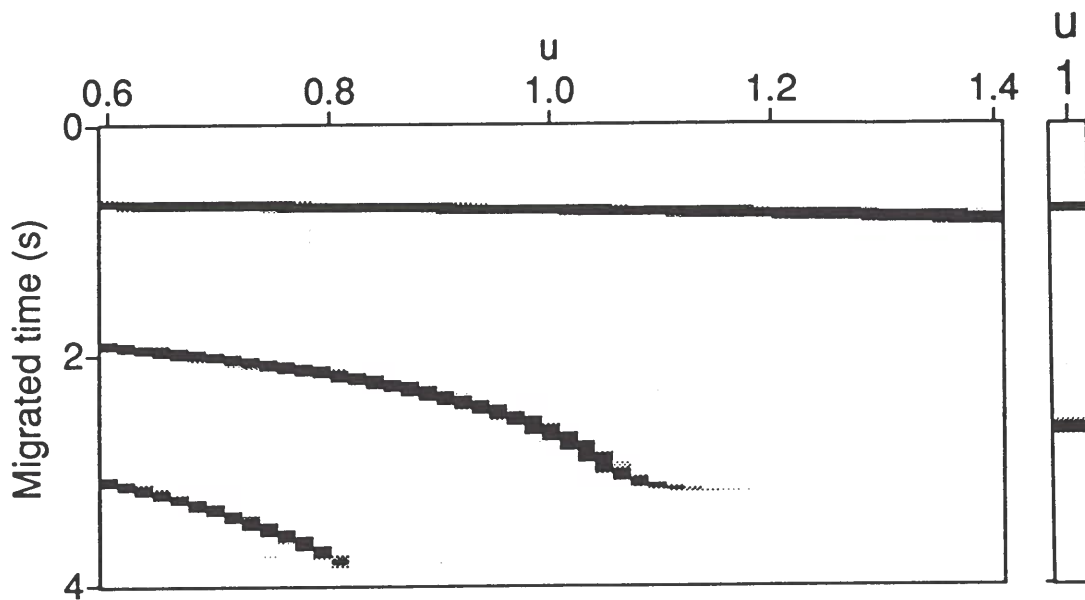

FIG. 16. A section sliced from the ensemble at $x = 5.2$ km, to be compared with the Gazdag migration at the right.
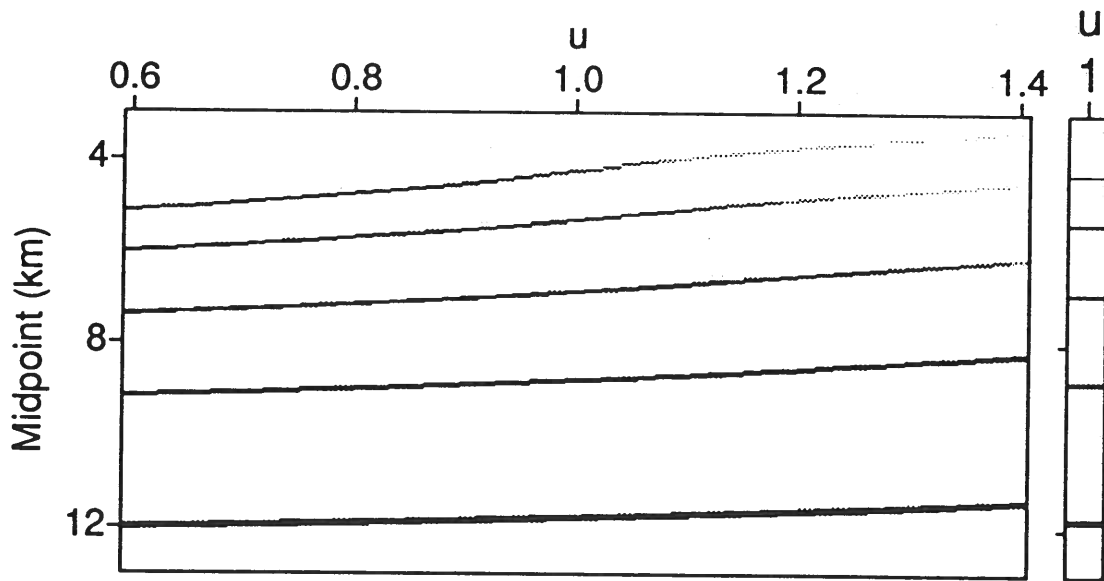
FIG. 17. A section sliced from the ensemble at $\tau = 3$ s to be compared with the Gazdag migration at the right.

Stolt-like migrations. This mapping is equivalent to using the correct dip-dependent imaging velocity of equation (8). This technique, which now properly handles ray bending where velocity varies with depth, retains the efficiency and flexibility of constant-velocity Stolt migration for both imaging and velocity estimation.

Specific conclusions

As with many migration algorithms, computation time increases with the dip to be properly imaged. Here, this cost is manifested as the need to prevent the aliasing of steeper events, which is done by assuring that the s axis is sampled finely enough. The computational effort required to perform Stolt-like ensemble migration is on the same order of creating an ensemble of constant-velocity Stolt migrations.

Stolt-like ensemble migration for numerous values of s is typically slower than a single Gazdag migration, so there is an initial investment to create the ensemble of migrated data. However, the ability to modify the velocity function, and indeed more accurately define the velocity function without having to re-migrate is advantageous. Also, within the context of time migration, lateral velocity variations can be tolerated, whereas they cannot be in a single Gazdag migration.

Recommendations

Li et al. (1991) and Schleicher et al. (1988) suggest the usefulness of prestack ensemble migration for velocity determination. The Stolt-like ensemble migration approach is readily extended to prestack migration as outlined in Appendix A.

ACKNOWLEDGMENTS

REFERENCES

Dix, C., 1955, Seismic velocities from surface measurements: Geophysics **20**, 68-86.

Fowler, P., 1984, Velocity independent imaging of seismic reflectors: Presented at the 54th SEG annual meeting, Atlanta.

Gazdag, J., 1978, Wave equation migration with the phase-shift method: Geophysics **43**, 1342-51.

Li, Z., Lynn, W., Chambers, R., Larner, K., Abma, R., 1991, Enhancements to prestack frequency-wavenumber (*f-k*) migration: Geophysics **56**, no. 1.

Schleicher, K., Grygier, D., and Iwamoto, T., 1988, Accurate migration velocities using prestack migration: Presented at the 58th SEG annual meeting, Anaheim.

Stolt, R. H., 1978, Migration by Fourier transform: Geophysics **43**, 23-48.
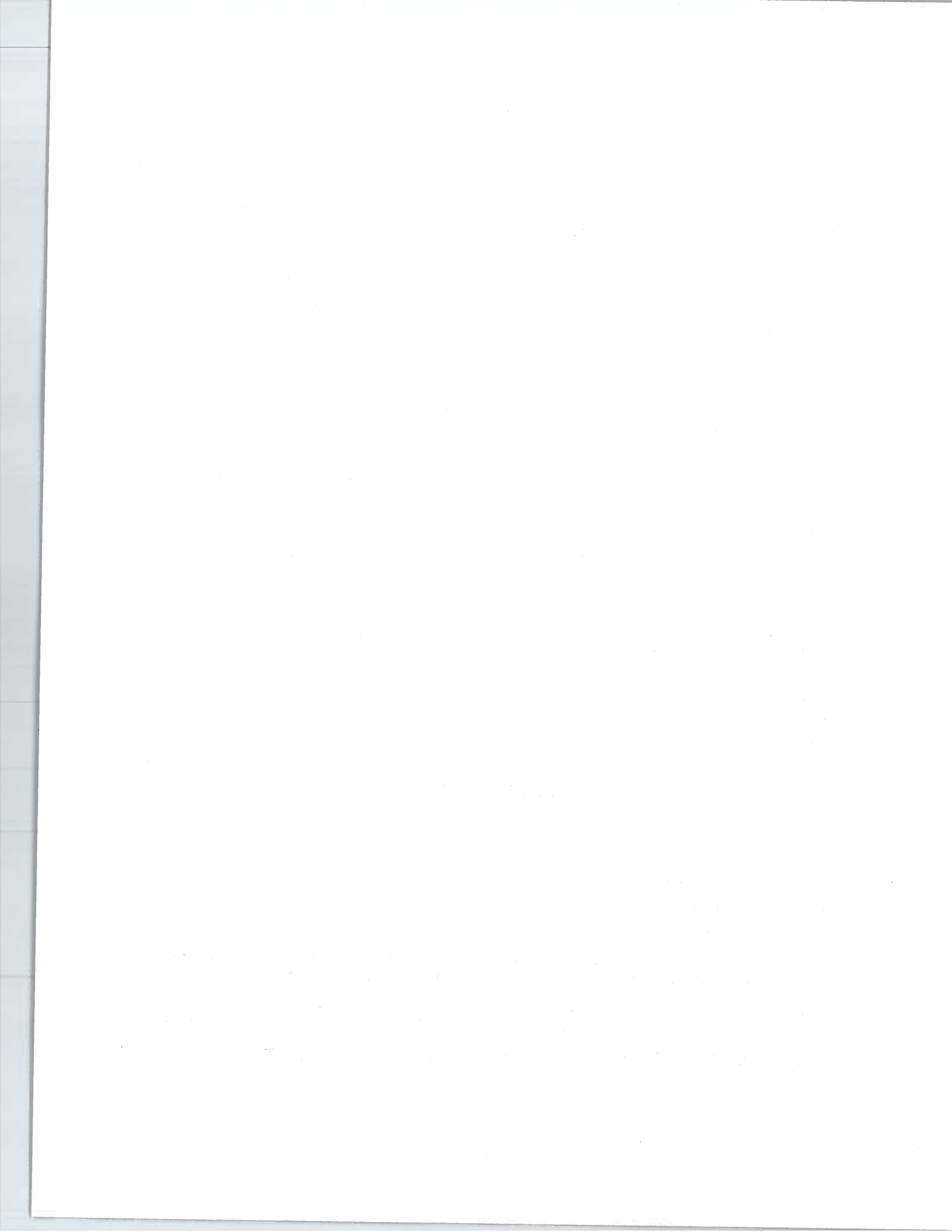
APPENDIX A

Prestack Stolt-like ensemble migration

The application of Stolt-like ensemble migration to stacked data has been demonstrated here. However, creating the ensemble of constant-velocity Stolt migrations is most often done before stack. Schleicher et al. (1988) and Li et al. (1991) use *prestack* Stolt migration to facilitate velocity estimation and imaging of steep dips. Because of this utility of prestack migration to determine velocity, it is valuable to derive prestack Stolt-like ensemble migration.

Prestack Gazdag migrated data q can be written as

$$q(\tau, x) = \frac{1}{8\pi^3} \int d\omega \int dk_h \int dk_x \, e^{ik_x x} \, e^{-i\omega \int_0^\tau d\sigma \left[\sqrt{1 - \frac{v^2(\sigma)(k_x - k_h)^2}{4\omega^2}} + \sqrt{1 - \frac{v^2(\sigma)(k_x + k_h)^2}{4\omega^2}} \right]} P(\omega, k_h, k_x),$$

where k_h is the wavenumber in the half-offset dimension, and k_x is the wavenumber in the midpoint direction (Stolt, 1978). P is therefore the recorded seismic wavefield transformed over recording time, half-offset, and midpoint.

The following change of variable makes the Gazdag equation look like a Fourier transform.

$$\omega_\tau \tau = \omega \int_0^\tau d\sigma \left[\sqrt{1 - \frac{v^2(\sigma)(k_x - k_h)^2}{4\omega^2}} + \sqrt{1 - \frac{v^2(\sigma)(k_x + k_h)^2}{4\omega^2}} \right]$$

By making the above change of variable, the migrated data become

$$q(\tau, x) = \frac{1}{8\pi^3} \int d\omega_\tau \int dk_h \int dk_x \, e^{-i\omega_\tau \tau} \, \tilde{Q}(\omega_\tau, \tau, k_h, k_x). \tag{20}$$

where

$$\tilde{Q}(\omega_\tau, \tau, k_h, k_x) = \left| \frac{\partial \omega}{\partial \omega_\tau} \right| P\left(\omega(\omega_\tau, \tau, k_h, k_x), k_h, k_x \right).$$

$\tilde{Q}$ and ω in this equation are functions of both ω_τ and τ. In order to avoid confusing this τ with the Fourier conjugate of ω_τ, replace τ with a different variable s, which yields

$$\tilde{Q}(\omega_\tau, s, k_h, k_x) = \left| \frac{\partial \omega}{\partial \omega_\tau} \right| P\left(\omega(\omega_\tau, s, k_h, k_x), k_h, k_x \right).$$

$\tilde{Q}$ has an inverse Fourier transform $\tilde{q}$, given by

$$\tilde{q}(\tau, s, k_h, k_x) = \frac{1}{2\pi} \int d\omega_\tau \, e^{i\omega_\tau \tau} \, \tilde{Q}(\omega_\tau, s, k_h, k_x).$$

Rewriting the migrated data in terms of these new variables,

$$\tilde{q}(\tau, s, x) = \frac{1}{8\pi^3} \int d\omega_\tau \int dk_h \int dk_x \, e^{-i\omega_\tau \tau} \, \tilde{Q}(\omega_\tau, s, k_h, k_x). \tag{21}$$

Again, by replacing τ by s in the argument of $\tilde{Q}$, the expression for desired migrated result $q(\tau, x)$ has been replaced by a Fourier transform. Thus, analogous to the post-stack case, when $s = \tau$, (or $u = 1$), $\tilde{q}(\tau, s = \tau, x) = q(\tau, x)$. Thus, a way of computing $q(\tau, x)$ is to perform a number of inverse Fourier transforms (using FFTs) for many different constant values of the parameter s, and subsequently carve out of this ensemble a section resulting in a prestack-migrated section. As before, remapping the data to the u ensemble most efficiently samples the ensemble.

APPENDIX B

Gazdag migration

To derive a Gazdag migration algorithm, begin with the wave equation, expressed by equation (22). Here, v is velocity, x and z are familiar orthogonal coordinates, t is time, and P is some solution to the partial differential equation. The following derivation is expressed in two dimensions; however, a full 3-D migration derivation is a simple matter of replacing $k_x x$ with $k_x x + k_y y$ in equation (23) and adding the partial derivative with respect to y in the wave equation below.

$$\frac{\partial^2 P}{\partial x^2} + \frac{\partial^2 P}{\partial z^2} = \frac{1}{v^2}\frac{\partial^2 P}{\partial t^2}. \tag{22}$$

A primitive solution of this equation is a plane wave, propagating forward in time and in the positive x and z directions. The angular frequency of the wave is represented by ω, and the wavenumbers in the x and z directions are given by k_x and k_z, respectively.

$$P = e^{ik_z z + ik_x x - i\omega t}. \tag{23}$$

By taking the appropriate derivatives and substituting them into the wave equation, the dispersion relationship is arrived at, shown in equation (24). The plane wave solution is acceptable only if this relationship between frequency and wavenumber is satisfied.

$$k_x^2 + k_z^2 = \frac{\omega^2}{v^2} \tag{24}$$

If equation (23) is a solution, any linear combination also must be a solution. Equation (25) demonstrates such a combination, with weights $A(k_x, \omega)$.

$$P(t, x, z) = \frac{1}{4\pi^2}\int dk_x \int d\omega\, A(k_x, \omega)\, e^{ik_z z + ik_x x - i\omega t}. \tag{25}$$

Now equation (24) is solved for k_z and substituted into equation (25). The sign convention of the square-root is established by the consideration of upgoing waves.

$$P(t, x, z) = \frac{1}{4\pi^2}\int dk_x \int d\omega\, A(k_x, \omega)\, e^{-i\sqrt{\frac{\omega^2}{v^2} - k_x^2}\, z + ik_x x - i\omega t}. \tag{26}$$

To obtain the weights $A(k_x, \omega)$, first set $z = 0$.

$$P(t, x, z = 0) = \frac{1}{4\pi^2}\int dk_x \int d\omega\, A(k_x, \omega)\, e^{ik_x x - i\omega t}.$$

Clearly, the weights are merely the Fourier transform of the data at the surface:

$$\begin{aligned} A(k_x, \omega) &= \int dx \int dt\, e^{-ik_x x + i\omega t}\, P(t, x, z = 0) \\ &= P(\omega, k_x, z = 0). \end{aligned} \tag{27}$$

Substitution of result (27) into equation (26) yields

$$P(t, x, z) = \frac{1}{4\pi^2} \int dk_x \int d\omega \, P(\omega, k_x, z = 0) \, e^{-i\sqrt{\frac{\omega^2}{v^2} - k_x^2} z + i k_x x - i\omega t}$$

So, to get the data at any depth z, merely do a phase shift on the data.

$$
\begin{aligned}
P(\omega, k_x, z) &= P(\omega, k_x, z = 0) \, e^{-iz\sqrt{\frac{\omega^2}{v^2} - k_x^2}} \\
&= P(\omega, k_x, z = 0) \, e^{-iz\frac{\omega}{v}\sqrt{1 - \frac{v^2 k_x^2}{\omega^2}}}
\end{aligned}
\tag{28}
$$

Gazdag migration is often written in vertical time rather than depth. Equation (28) then becomes

$$P(\omega, k_x, \tau) = P(\omega, k_x, \tau = 0) \, e^{-i\omega\tau\sqrt{1 - \frac{v^2 k_x^2}{\omega^2}}}.$$

By integrating over frequency and allowing velocity to vary with two-way vertical time τ, an equation for the migrated data q is obtained. Since the equation is in time rather than depth, k_z no longer appears in these equations. Therefore, for the sake of simplicity, the x subscript on k_x has been dropped; k now unambiguously means horizontal wavenumber.

$$q(\tau, k) = \frac{1}{2\pi} \int d\omega \, e^{-i\omega \int_0^\tau d\sigma \sqrt{1 - \frac{v^2(\sigma)k^2}{\omega^2}}} \, P(\omega, k). \tag{29}$$

APPENDIX C

The Cauchy inequality

To show that the second term in equation (9) is never negative, consider the Cauchy inequality,

$$\int_\alpha^\beta d\sigma\, g\, h \leq \sqrt{\int_\alpha^\beta d\sigma\, g^2 \int_\alpha^\beta d\sigma\, h^2},$$

which holds true for any real functions g and h. By letting $g = 1$, $h = v^2$, $\alpha = 0$, and $\beta = \tau$,

$$\int_0^\tau d\sigma\, v^2 \leq \sqrt{\tau \int_0^\tau d\sigma\, v^4}.$$

Squaring both sides and dividing through by τ^2,

$$\frac{1}{\tau^2}\left(\int_0^\tau d\sigma\, v^2\right)^2 = v_{rms}^4 \leq \frac{1}{\tau}\int_0^\tau d\sigma\, v^4.$$

Thus, the second term in equation (9) is never negative for any velocity function $v(\tau)$.
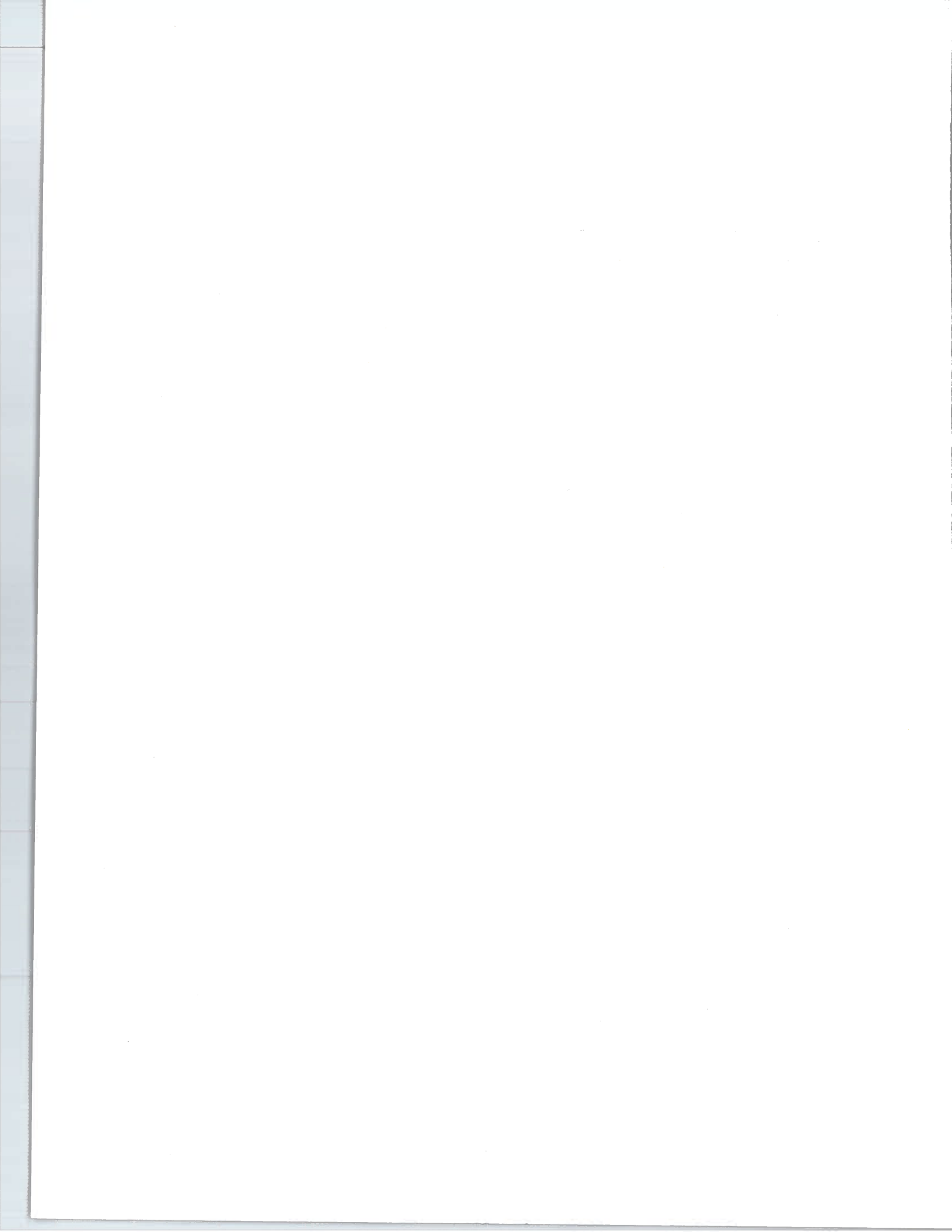
APPENDIX D

Stolt-like ensemble migration

```c
/* This program does v(t) migration via a Stolt-like ensemble

        Program ensmig.c by Weston Mikulich
        Written on June 8, 1990
        Last modified Wednesday, August 1, 1990
*/

#include "par.h"

/* prototypes for functions defined and used below */
void stm1k (float k,
        int nt, float dt, float ft,
        int ntau, float dtau, float ftau,
        float v, complex *p, complex *q, float **wtow, complex **r,
        int nttab, int nu, float du, float fu, float pmax, int nptab);
complex csinci (float xout, int nxin, float dxin, float fxin, complex *yin);
float velfn(float t);
void wtable(int nptab, int nttab, float lt, float **wtow, float pmax);
void getvec(complex ***data, complex **cq, int iu, int nt, int nk);

/* the main program */
main ()
{
        int nt,ntau,nx,ix,it,itau,nxfft,nk,ik, nptab,nttab,iu,nu,nwtau;
        float dt,ft,dtau,ftau,dk,fk,k,dx,v,**p,**q, **wtow, pmax,lt,u,du,fu,**error;
        complex **cp,**cq, **r,***data;

        /* get optional parameters */
        nt = 293; dt = 0.02743; ft = 0.0; ntau = nt; dtau = dt; ftau = ft;
        nx = 514; dx = 0.032; nttab = 100; nptab = 50;
        fu = 0.8; nu = 12; du = 0.04;

        /* determine wavenumber sampling (for real to complex FFT) */
        nxfft = npfar(nx);
        nk = nxfft/2+1;
        dk = 2.0*PI/(nxfft*dx);
        fk = 0.0;
        nwtau = npfa(ntau);

        lt = ft+(nt-1)*dt;
        pmax = 2/velfn(0);

        /* allocate space */
        p = alloc2float(nt,nxfft);
        q = alloc2float(ntau,nxfft);
        cp = alloc2complex(nt,nk);
        cq = alloc2complex(ntau,nk);
        wtow = alloc2float(nttab,nptab+1);
        r = alloc2complex(ntau,nu);
        data = alloc3complex(ntau,nu,nk);
        error = alloc2float(nwtau,nttab);

        /* make table of wtau over w (s, p) */
        wtable(nptab, nttab, lt, wtow, pmax);

        /* read in zero-offset data */
        fread(p[0],sizeof(float),nt*nx,stdin);

        /* pad with zeros and Fourier transform x to k */
        for (ix=nx; ix<nxfft; ix++)
                for (it=0; it<nt; it++)
                        p[ix][it] = 0.0;
        pfa2rc(-1,2,nt,nxfft,p[0],cp[0]);

        /* migrate each wavenumber */
        for (ik=0,k=fk; ik<nk; ik++,k+=dk) {
                stm1k(k,nt,dt,ft,ntau,dtau,ftau,v,cp[ik],cq[ik],
```

```
                                wtow, r,nttab,nu,du,fu,pmax,nptab);
                for (iu=0; iu<nu; iu++) {
                        for (it=0; it<nt; it++) {
                                data[ik][iu][it] = r[iu][it];
                        }
                }
                fprintf(stderr,"percent done = %i \n", ik*100/nk);
        }

        /* For all u, FFT k to x */
        for (iu=0; iu<nu; iu++) {
                /* Fourier transform k to x (including FFT scaling) */
                getvec(data,cq,iu,nt,nk);
                pfa2cr(1,2,ntau,nxfft,cq[0],q[0]);
                for (ix=0; ix<nx; ix++)
                        for (itau=0; itau<ntau; itau++)
                                q[ix][itau] /= nxfft;

                /* write migrated data(t)(x)(u) */
                fwrite(q[0],sizeof(float),ntau*nx,stdout);
        }
}

void stm1k (float k,
        int nt, float dt, float ft,
        int ntau, float dtau, float ftau,
        float v, complex *p, complex *q, float **wtow, complex **r,
        int nttab, int nu, float du, float fu, float pmax, int nptab)
/* migrate one wavenumber k */
{
        int ntfft,ntaufft,nw,nwtau,it,itau,iw,iwtau,
                is,indp,ip,iu;
        float dw,dwtau,fw,fwtau,lt,w,wtau,phase,scale,
                s,ps,T,dp,*wtaua,*wa,*sarray,u,tau,xsampl,pmaxs;
        complex comzero,*pp,*qq,**qs, *values,*phat;

        /* determine last time */
        lt = ft+(nt-1)*dt;

        /* determine input and output frequency sampling */
        ntfft = npfa(nt);
        nw = ntfft;
        dw = 2.0*PI/(ntfft*dt);
        fw = -PI/dt;
        ntaufft = npfa(ntau);
        nwtau = ntaufft;
        dwtau = 2.0*PI/(nwtau*dtau);
        fwtau = -PI/dtau;
        dp = pmax/(nptab-1.0);
        comzero = cmplx(0,0);

        /* allocate workspace */
        pp = alloc1complex(nw);
        qq = alloc1complex(nwtau);
        wtaua = alloc1float(nw*4);
        wa = alloc1float(nwtau+1);
        qs = alloc2complex(ntau,nttab);
        phat = alloc1complex(nttab);
        sarray = alloc1float(nu);
        values = alloc1complex(nu);

        /* pad with zeros and Fourier transform t to w, with w centered */
        for (it=0; it<nt; it++)
                pp[it] = (it%2 ? cneg(p[it]) : p[it]);
        for (it=nt; it<ntfft; it++)
                pp[it] = cmplx(0.0,0.0);
        pfacc(1,ntfft,pp);

        /* phase shift since time axis is not centered */
        for (iw=0,w=fw; iw<nw; iw++,w+=dw) {
                phase = w*ft-w*(ft+lt)/2.0;
                pp[iw] = cmul(pp[iw],cmplx(cos(phase),sin(phase)));
        }
```

```
/* loop over all s, as in table */
for (is=0, s=0.0; is<nttab; s+=lt/(nttab-1), is++) {

        v = velfn(s);
        pmaxs = 2/v;

        /* loop over frequency w */
        for (iw=0,w=0.0001; iw<nw*4; iw++,w+=dw/4) {

                /* compute array of wtau(w); go past evanescent
                   to edge of wtow table */
                ps = k/sqrt(w*w);
                if (ps>pmax) ps = pmax;
                indp = ps/dp;
                /* linearly interpolate table along p */
                T = wtow[indp+1][is] - (dp*(indp+1) - ps)*
                        (wtow [indp+1][is]- wtow[indp][is])/dp;
                wtaua[iw] = w * T;
        }

        /* invert to get w(wtau) by using y(x) to x(y) */
        yxtoxy(nw*4,dw/4,0.0001,wtaua,nwtau,dwtau/2,dwtau/2,0,w,wa);

        /* loop over migrated frequency wtau */
        for (iwtau=0,wtau=-nwtau*dwtau/2; iwtau<nwtau;
                iwtau++,wtau+=dwtau) {

                /* compute w = w(wtau) at which to evaluate p(w) */
                if (wtau<0.0) {
                        w = wa[(nwtau-1)-iwtau*2];
                        w = -w;
                } else {
                        w = wa[iwtau*2 - (nwtau+1)];
                }

                /* special case at the surface */
                if (is < 1) {
                        w = sqrt(pow(wtau,2.0)+pow(0.5*v*k,2.0));
                        if (wtau<0.0) w = -w;
                }

                /* can't handle really small wtau */
                if (sqrt(wtau*wtau) < .01)
                        w = sqrt(v*v*k*k/4);

                /* compute q(wtau) = p(w(wtau)) */
                qq[iwtau] = csinci(w,nw,dw,fw,pp);

                /* phase shift since migrated time axis is not centered */
                phase = -wtau*ftau+w*(ft+lt)/2.0;
                qs[is][iwtau] = cmul(qq[iwtau],
                        cmplx(cos(phase),sin(phase)));
        }

        /* Fourier transform q(wtau) to q(tau), accounting for scaling */
        pfacc(-1,ntaufft,qs[is]);
        scale = 1.0/ntaufft;
        for (itau=0; itau<ntau; itau++) {
                qs[is][itau] = (itau%2 ? cneg(qs[is][itau]) : qs[is][itau]);
                qs[is][itau] = crmul(qs[is][itau],scale);
        }
}

/* Map q[s][t] to r[u][t]; u=s/tau */
for (itau=0, tau=ft; itau<ntau; itau++, tau+=dt) {
        for (iu=0, u=fu; iu<nu; iu++, u+=du) sarray[iu] = u*tau;
        for (is=0; is<nttab; is++) phat[is] = qs[is][itau];

        /* interpolate */
        xsampl = lt/(nttab-1.0);
        ints8c(nttab,xsampl,0.0,phat,comzero,comzero,nu,sarray,values);
        for (iu=0; iu<nu; iu++) r[iu][itau] = values[iu];
}
```

```
                /* free workspace */
                free1complex(pp);
                free1complex(qq);
                free1float(wtaua);
                free1float(wa);
                free2complex(qs);
        }


        complex csinci (float xout, int nxin, float dxin, float fxin, complex yin[])
        /* tapered sinc interpolation of uniformly-sampled complex function y(x) */
        {
        #define LHALF 4
                int ixin,ixinlo,ixinhi;
                float xoutn,arg,taper;
                complex yout;

                /* initialize output */
                yout = cmplx(0.0,0.0);

                /* normalize xout */
                xoutn = (xout-fxin)/dxin;

                /* determine range of input samples that contribute */
                ixinlo = MAX(0,(int)(xoutn)-LHALF+1);
                ixinhi = MIN(nxin-1,(int)(xoutn)+LHALF);

                /* loop over input samples that contribute to output sample */
                for (ixin=ixinlo; ixin<=ixinhi; ixin++) {
                        arg = xoutn-ixin;
                        taper = 0.54+0.46*cos(arg*PI/LHALF);
                        yout = cadd(yout,crmul(yin[ixin],fsinc(arg)*taper));
                }

                return yout;
        }


        float velfn(float t)
        /* This returns the velocity at two-way zero-offset time t */
        {
                float a, b;
                a=1.6; b=0.5;
                return a*exp(b*t/2);
        }

        void wtable(int nptab, int nttab, float lt, float **wtow, float pmax)
        /* This function computes the table of values for
        omegatau over omega as a function of p (slope) and t. */
        {
                int ip, is, it;
                float p, s, sum, t, v;

                /* calculate wtau/w for every p and t */
                for (ip=0, p=0.0; ip<nptab+1; p+=pmax/(nptab-1), ip++) {
                        for (it=0, t=0.0; it<nttab; t+=lt/(nttab-1), it++) {
                                if (p<.01 || t<0.001) {
                                        wtow[ip][it] = 1.0;
                                } else {
                                        sum = 0.0; is = 0;
                                        if ( p < 2/velfn(t) ) {
                                                for (s=0.0; s<t+.001; s+=0.004) {
                                                        v = velfn(s);
                                                        is++;
                                                        sum += sqrt(1.0-(v*v*p*p/4));
                                                }
                                                wtow[ip][it] = sum/is;
                                        } else { wtow[ip][it] = .001; }
                                }
                        }
                }
        }


        void getvec(complex ***data, complex **cq, int iu, int nt, int nk)
        /* This returns the vector to be IFFT'd */
```
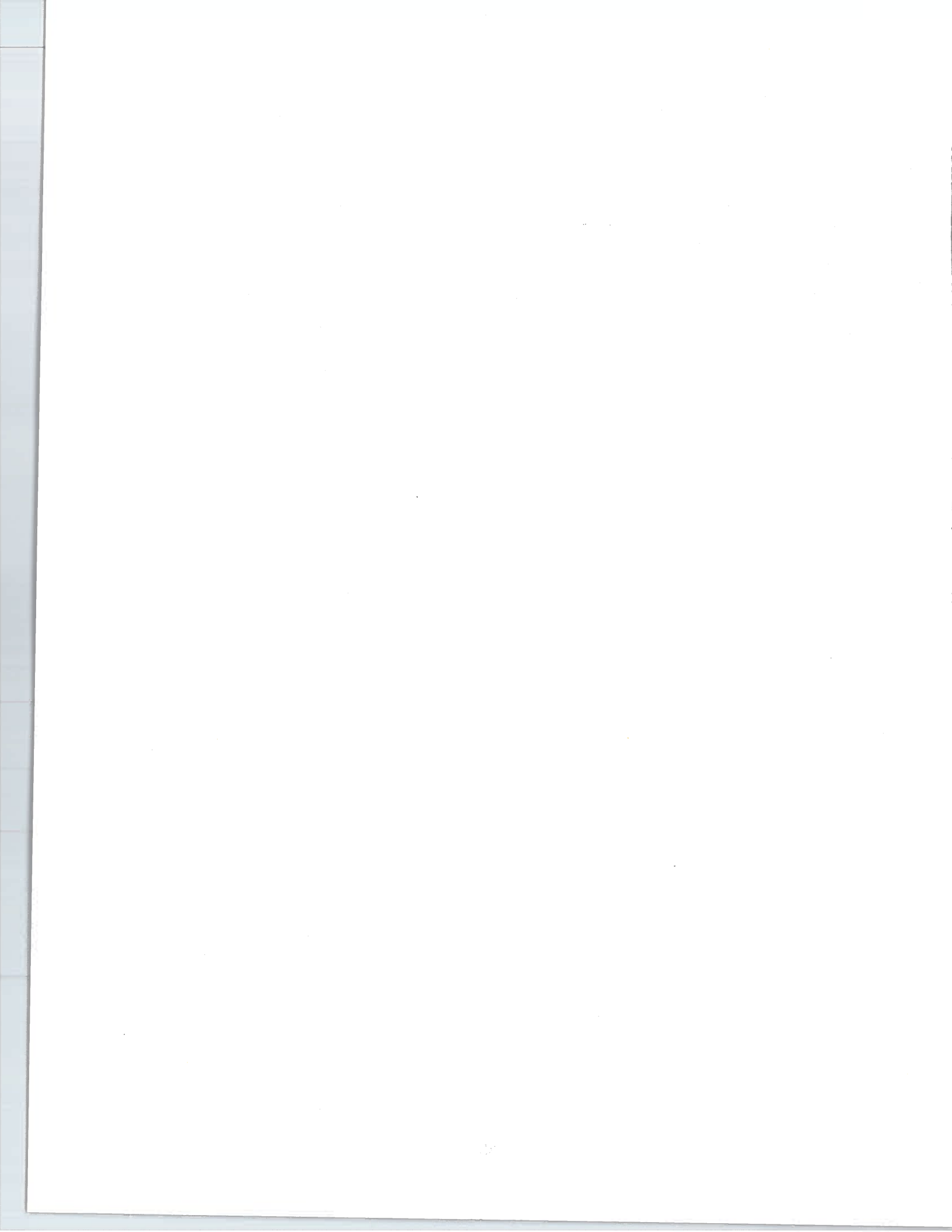
```
{
        int it, ik;

        for (it=0; it<nt; it++) {
                for (ik=0; ik<nk; ik++) {
                        cq[ik][it] = data[ik][iu][it];
                }
        }
}
```

GLOSSARY

symbol	definition	first use
A	weighting factors in plane-wave superposition	Equation (25)
α	arbitrary lower limit of integration	Appendix C
b	velocity gradient	page 16
$\tilde{b}$	incorrect velocity gradient	page 16
β	arbitrary upper limit of integration	Appendix C
δ	delta function	page 5
f	frequency	Equation (7)
g	arbitrary function of σ	page 16
h	arbitrary function of σ	Appendix C
i	square root of -1	Equation (3)
k	horizontal wavenumber; k_x	Equation (3)
k_h	wavenumber in half-offset direction	Appendix A
k_x	horizontal (midpoint) wavenumber	Equation (23)
k_y	wavenumber in y direction	Appendix B
k_z	vertical wavenumber; conjugate variable of z	Equation (23)
P	Fourier-transformed seismic data	Equation (3)
p	ray parameter; slope in $k - \omega$ domain	Equation (7)
$\tilde{Q}$	Fourier transform of migrated data	Equation (12)
q	migrated data	Equation (3)
s	parameter at which $\tilde{Q}$ is a Fourier transform	Equation (13)
σ	dummy variable of time integration	Equation (1)
σ'	dummy variable equal to $u\,\sigma$	page 16
t	two-way traveltime	Equation (1)
τ	vertical two-way (migrated) time	Equation (1)
u	s/τ	page 14
$v(\sigma)$	instantaneous velocity as a function of vertical time	Equation (1)
$\tilde{v}(\sigma)$	incorrect velocity function	Equation (18)
v_{rms}	root-mean-square velocity	Equation (2)
v_s	imaging velocity	Equation (4)
ω	unmigrated (angular) frequency	Equation (3)
ω_τ	migrated frequency	Equation (5)
$\tilde{\omega}_\tau$	migrated frequency from incorrect velocities	Equation (18)
x	horizontal distance	Equation (1)
y	horizontal distance perpendicular to x	Appendix B
z	depth	Equation (22)