A reversible transform for seismic data processing

William A. Burnett* and Robert J. Ferguson

ABSTRACT

We use the nonstationary equivalent of the Fourier shift theorem to derive a general one-dimensional integral transform for the application and removal of certain seismic data processing steps. This derivation comes from the observation that many seismic data processing steps can be viewed as nonstationary shifts themselves. The nonstationary frequency and time distortions inherent to many seismic data processes are predicted and quantified by this transform. Once the transform is developed, we demonstrate matrix-vector multiplication as a valuable tool for implementation.

INTRODUCTION

It is well known that the Fourier transform has a variety of properties that make it useful for signal processing, (Yilmaz, 2001). The Fourier transform essentially rearranges the data in an input function into its complex frequency components, (Karl, 1989). This can be done because continuous functions and their discretely sampled equivalents can be exactly expressed in terms of the Fourier basis. Once the data are rearranged in terms of the Fourier basis, the function is said to be in the Fourier domain. The organization of data in the Fourier domain allows many otherwise complex operations to be applied easily. Properties such as the Faltung, Wiener-Khintchine and shift theorems, (Sneddon, 1995) exploit this organization, and they are allowed to exploit it because the Fourier transform is exactly reversible.

The inversion theorem of the Fourier transform states that no information is lost when a function is moved in and out of the Fourier domain, (Sneddon, 1995). This property is critical in quantitatively justifying valid processing with the Fourier transform. The Fourier transform can be applied to an input signal, then the signal can be manipulated, and then an inverse Fourier transform can then be applied without any residual effects of the transformation itself. Only the effects of the manipulation are present in the output signal.

Viewing the Fourier transform from this perspective suggests that any quantitatively valid processing domain should be reached by an exactly reversible transform. There are many transformations that are exactly reversible, but what makes the Fourier transform so useful to geophysicists is that the organization of the data in the Fourier domain is easy to exploit for desirable processing steps. Therefore, a useful transform will not only be reversible, but it should also rearrange the input data into an exploitable organization.

Many individual seismic data processing steps also simply rearrange input data. Barring destructive processes such as muting or band-limited frequency filtering, even an entire data processing flow is itself simply a rearranging of the input data set. Processing steps and flows that rearrange data are essentially performing a transformation of that data into a

*University of Texas, Austin

corrected set. However, conventional methods used to perform processing steps of this type are usually not implemented as transforms. Instead, they are implemented with data-mapping algorithms that require interpolation, (Harlan, 1982).

Most conventional interpolation schemes contradict the primary assumption of the Fourier transform, as they do not assume that the continuous equivalent of the recorded data is constructed with a Fourier basis. Instead, they assume some other basis in order to design efficient, but approximate interpolants. These methods not only have the risk of immediately altering the data, but are also inherently irreversible. Applying or removing processing steps causes a small amount of information loss. With quantitative analysis methods as a goal in seismic data processing, it is important to preserve as much of the input information as possible during the flow.

In order to address the problem of data loss due to interpolation-based methods, we propose a reversible one-dimensional transform for the implementation of seismic data processing steps. Implementing data processing steps as reversible transforms does not require interpolation in the continuous case, and correctly assumes the Fourier basis in agreement with the Fourier transform for interpolation in the discrete case. Since the Fourier basis is used, the forward transform is theoretically exactly invertible. Just as the inversion theorem quantitatively justifies the Fourier domain as a valid processing domain, a reversible transform for data processing steps justifies the use of corrected datasets as if they are in valid processing domains themselves. Although several seismic data processing operations have been individually described by transforms of exactly this type ((Margrave, 1998), (Margrave and Ferguson, 1999), (Margrave, 2001)), the transform here is the general form. Any processing step that can be viewed as a nonstationary shift can be easily implemented by this transform. Seismic data processing steps are naturally useful for separating signal from noise, so they offer familiar, exploitable organizations of data. Therefore, a reversible transform for seismic data processing offers a useful set of quantitatively valid domains in which to work.

The general data processing transform we derive is based on classical Fourier transform theory ((Papoulis, 1962), (Sneddon, 1995)) and the theory of nonstationary filtering, (Margrave, 1998). The development of a reversible transform comes directly from filtering properties of the Fourier transform, so the parallels are straightforward. In fact, the reversible data processing transform proposed here is a special case of the Fourier shift theorem adapted to nonstationary filtering.

THEORY

The Fourier transform obeys the inversion theorem, which states that it is exactly reversible. Because of this, data can be moved in and out of the Fourier domain with no information loss. This justifies the Fourier domain as a quantitatively valid domain in which to process data. Many conventional processing methods are not exactly reversible, so an improvement upon them would be to find an equivalent, but reversible processing method. We now develop a general form for a forward transform that applies data processing steps which can be formulated as nonstationary shifts. Following this, we derive the inverse of the transform.

It is important to have already defined a convention for the forward and inverse Fourier transform before proceeding. The shift-direction determined by the sign in the shifting exponential will reverse if the opposite sign convention for the Fourier transform is used. With the Fourier transform convention defined in equation 1, a positive sign in the shifting exponential causes a backwards shift of the input function, and a negative sign in the shifting exponential causes a forward shift.

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-\mathrm{i}\omega \mathrm{t}} dt, \tag{1}$$

Keeping track of the signs in the exponents is critical in developing the following transform.

Approximate nonstationary shifts implemented in the time domain are constrained to evaluate the forward and reverse process using only the input grid. As an example, the mechanics of the shift required for Normal Move Out (NMO) simply shifts data values backwards from input time, t_x, to output time, t_0, where t_x depends on t_0. This is exactly a nonstationary shift by combination as described by Margrave (1998), where for NMO correction, the nonstationary shift, is expressed as,

$$\Delta_{\mathrm{NMO}}\left(t_0\right) = t_x\left(t_0\right) - t_0. \tag{2}$$

Figure 1 shows what this shift simply represents when looking at a shot or common midpoint (CMP) gather. Other data processing steps have similar mechanics. In general, any processing step that requires a mapping from an input coordinate that is a function of the output coordinate, to that output coordinate, can be viewed as a nonstationary shift by combination. It is better at this point to think in terms of input and output coordinates rather than input and output times, because then even frequency, wavenumber, space, or time shifts can be admitted, depending on the desired domain. To emphasize that the input and output coordinates do not necessarily correspond to time, we use the following notation for a general nonstationary data processing shift,

$$\Delta(q) = p(q) - q, \tag{3}$$

where *p* and *q* are the input and output coordinates, respectively. Using these general coordinates, the reverse nonstationary processing shift is,

$$h(q) \equiv f(q + \Delta(q)) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\beta) e^{\mathrm{i}\beta\Delta(q)} e^{\mathrm{i}\beta \mathrm{q}} d\beta, \tag{4}$$

where β is the Fourier dual of *q*. The convenient form of Δ*(q)* allows the exponentials to be reduced, giving a concise form of the forward data processing transform,

$$h(q) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\beta) e^{\mathrm{i}\beta \mathrm{p}(q)} d\beta, \tag{5}$$

which confirms the desired result,

$$h(q) = f(p). \tag{6}$$

The output function, *h(p)*, is a processed signal, so for example, if Δ_{NMO} is substituted into equation 4, then *h(q)* would be equivalent to *f(q)* with the NMO correction applied.

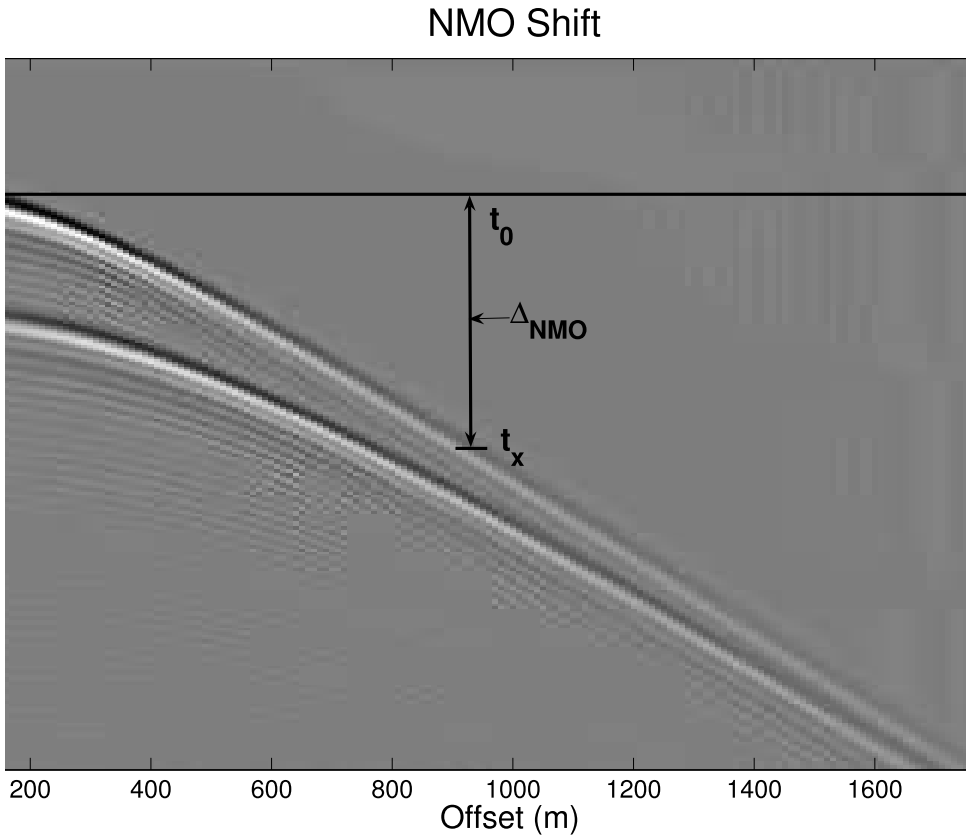

FIG. 1. Synthetic shot gather showing the simple mechanical meaning of Δ_{NMO}. This value is nonstationary because it changes depending on not only the offset, but also on the time assumed to be t_0.

This agrees with expression 6 as the value of the output function *h* at the output time, *q*, is the same as the value at the input coordinate, *p*, in the input function, *f*.

The forward transform given here is in the mixed-domain. For example, if the input function is defined over time, it takes the frequency spectrum of that input function, and outputs a new time series. This should not be alarming, as the Fourier transform itself is a mixed-domain transform. It may seem questionable how *p(q)* is handled in the exponent. However, since the nonstationary shift is formulated here as a nonstationary combination, and the integration is over β, nothing new is needed to handle the relation between *p* and *q*. Nonstationary combination is a filtering process that, although nonphysical in most cases, provides exactly the type of shift desired for the NMO correction. The distortion of the input signal is handled by the inverse transform developed below, which relies on physically correct nonstationary convolution instead.

To derive the inverse of this transform, it helps to recognize what is already known about the desired output trace. First of all, notice that the forward transform shifts the data while simultaneously performing the *inverse* Fourier transform. It seems reasonable to expect the inverse transform to start with the corrected trace, and shift it in the opposite direction as it performs the *forward* Fourier transform. That is, the inverse transform naturally should recover the original input spectrum, *F(β)*, rather than go directly back to the original input trace.

The output function of the forward transform, *h(p)*, becomes the input function of the inverse transform. Since the nonstationary behaviour still depends on *q*, which is now the input coordinate, the inverse transform is a nonstationary convolution operator. Although nonstationary combination mechanically performs nonstationary shifts correctly, amplitudes are not changed, and therefore energy is not conserved between input and output traces. Figure 2 demonstrates how the area under the input signal changes under a nonstationary combination shift. Rayleigh's theorem (also known as Plancherel's theorem) states that the integral of the square of an input function must be the same as the integral of its squared Fourier transform, (Karl, 1989). Physically, this means that energy should be conserved between the input and Fourier domains, and clearly, the forward data processing transform violates this. The inverse data processing transform is based on the physically valid filtering process of nonstationary convolution, and can be used to account for changes in energy caused by the forward transform.

Rayleigh's theorem relates the energy of a function to the area beneath the function. To predict how the energy has been changed by the forward transform, take the ratio of each shifted rectangular element as in figure 2:

$$\alpha = \frac{\Delta \mathrm{p} \left(\frac{f(p_1)+f(p_2)}{2} \right)}{\Delta \mathrm{q} \left(\frac{h(q_1)+h(q_2)}{2} \right)}. \tag{7}$$

Equation 6 shows that the shifted values of the input function are not changed. This corresponds to the height of the rectangular elements shown in figure 2 remaining unchanged

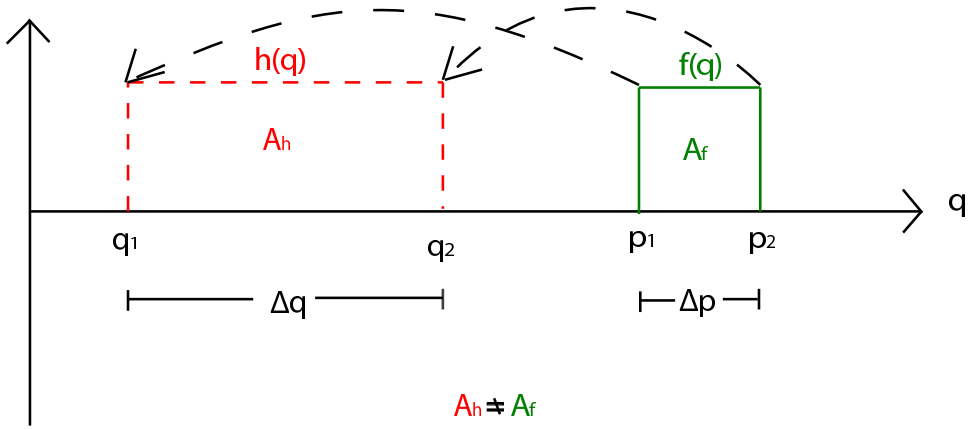

FIG. 2. Element area change caused by a nonstationary combination shift. The values of *f* at p_1 and p_2 are mapped to *h* at q_1 and q_2, respectively. Since Δp does not equal Δq, the areas are not equal, so by Rayleigh's Theorem, energy is not conserved. Scaling *h(q)* by $\alpha(q)$ makes the areas the same.

by the forward transform, giving,

$$\alpha \equiv \frac{\Delta \mathrm{p}\left(\frac{f(p_1)+f(p_2)}{2}\right)}{\Delta \mathrm{q}\left(\frac{f(p_1)+f(p_2)}{2}\right)} = \frac{\Delta \mathrm{p}}{\Delta \mathrm{q}}. \tag{8}$$

In the continuous limit, 8 goes to,

$$\alpha = \frac{\partial p}{\partial q}. \tag{9}$$

The output function, *h(q)*, is of course distorted by different amounts since the applied shift is nonstationary. α does not contradict this, as for nonstationary shifts, *p* is a function of *q*, and so α itself is also a function of *q*. By scaling $h(q)$ by $\alpha(q)$ and integrating, we claim that the energy change is accounted for.

Now that there is a mechanism for removing amplitude distortions, the rest of the inverse transform development comes from removing the mechanical shift. The familiar approach to un-shifting is to apply the Fourier transform to *h(q)* in terms of its own coordinate, *q*, then apply the nonstationary shift, and finally apply the inverse Fourier transform:

$$y(q) \equiv h(q - \Delta(q)) = \frac{1}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(q)e^{-\mathrm{i}\beta \mathrm{q}}dq e^{-\mathrm{i}\beta\Delta(q)}e^{\mathrm{i}\beta \mathrm{q}}d\beta. \tag{10}$$

Moving the shifting exponential into the forward Fourier transform integral gives,

$$y(q) = \frac{1}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(q)e^{-\mathrm{i}\beta \mathrm{q}}e^{-\mathrm{i}\beta\Delta(q)}dq e^{\mathrm{i}\beta \mathrm{q}}d\beta. \tag{11}$$

Now again exploit the convenient form of $\Delta(q)$ to reduce the exponentials to a single kernel for the inner integral,

$$y(q) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h(q) e^{-\mathrm{i}\beta \mathrm{p}(q)} dq e^{\mathrm{i}\beta \mathrm{q}} d\beta, \tag{12}$$

but this expression forces integrating *p(q)* in the exponential over *q*. This may at first seem like a less attractive route than the familiar approach, but it can be used to account for the nonphysical distortion caused by the forward transform by nonstationary combination. Remember from figure 2, and equation 9, that the energy change can be accounted for by scaling *h(q)* by α inside an integral over *q*. Now remember that *h(q)=f(p)*, and then the scaling factor, α, conveniently acts as a change-of-integration-variable factor, giving,

$$y(q) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \alpha \mathrm{f}(p) e^{-\mathrm{i}\beta \mathrm{p}(q)} dq e^{\mathrm{i}\beta \mathrm{q}} d\beta. \tag{13}$$

Including α allows the inner integral to be evaluated in the same form as a Fourier transform. Now notice that the outer integral in equation (13) is exactly the inverse Fourier transform of the original uncorrected spectrum. Therefore, the mixed-domain general inverse data processing transform is given by,

$$F(\beta) = \int_{-\infty}^{\infty} \mathrm{f}(p) e^{-\mathrm{i}\beta \mathrm{p}(q)} dp. \tag{14}$$

The uncorrected trace can be exactly recovered by a regular inverse Fourier transform:

$$y(q) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\beta) e^{\mathrm{i}\beta \mathrm{q}} d\beta. \tag{15}$$

The desired result of the inverse transform is obtained:

$$y(q) = f(q). \tag{16}$$

Although the mechanics of this general data processing transform are clear under nonstationary filtering theory, we demonstrate that for any data processing step implemented by transform, α trivially predicts amplitude corrections that are otherwise difficult to quantify under conventional stationary filtering theory.

IMPLEMENTATION

Now that the general form of the transform has been developed, an appropriate method is needed to implement it. The method we use and discuss here is matrix vector multiplication.

In order to computationally implement an integral transform, recognize that the input function can be viewed as a vector, or a one-dimensional array, and that the transform kernel can be viewed as a matrix, or a two-dimensional array. The output of a matrix vector multiplication is analogous to the output function of the integral, but it is another vector along a new axis. The important feature of matrix vector multiplication is that it can be used to simultaneously multiply and integrate two functions over an input index. This concept is certainly not new to geophysics, and many of its benefits have been explored in detail, (Claerbout, 1992).

For any transform, the size of its matrix is dependent on the length of the input vector. For an input vector of *N* elements, the transform matrix will be of size $N \times N$. The general structure of the matrix is always the same though, in that the horizontal index, or column number, corresponds to the input coordinate, and the vertical index, or row number, corresponds to the output coordinate. In the case of seismic data, an input vector would typically be a seismic trace, but the input vector can be a discrete frequency or wavenumber spectrum as well. To start though, take for example a trace that was recorded starting at *t=0* for three seconds with a sampling interval of 4 ms. This trace would have 751 samples, and would therefore be a vector of length *N*=751. The full form of a transform matrix suitable to operate on this trace must be of size *NxN*. It is not uncommon to reduce the size of a transform or operator matrix to increase computational efficiency, but this cannot be justified without first observing symmetries of the full form. Increasing the size of the matrix is allowed and helpful in some cases. This is done by zero-padding (appending rows or columns of zeros), but this generally does not add any new information.

As usual, we use the Fourier transform to demonstrate the general concepts for constructing transform matrices. In the case of the Fourier transform, each horizontal index of the matrix corresponds to an input time increment. Each vertical index corresponds to an output frequency increment. The Fourier transform can be written as a matrix vector multiplication according to (Karl, 1989, ch. 5):

$$F(\omega_j) = \sum_{l=1}^{N} e^{-\mathrm{i}\omega_j t_l} f(t_l). \tag{17}$$

An amazing but often forgotten feature of the discrete Fourier transform is that it can actually output a valid *continuous* frequency spectrum, (Gubbins, 2004). This implies that the frequency axis must be chosen in order to precisely implement the Fourier transform. As mentioned above, no new information can be added by increasing the number of output values. Conservation of information is a key idea throughout transform theory, and it comes up again when remembering that the output of the Fourier transform is complex. Since each output value actually has two parts (real and imaginary), the output list of unique values should be only about half the length of the input list. Also, realizing that the Nyquist frequency is the highest recoverable frequency from an input signal, the optimal Fourier transform only outputs *N*/2+1 frequency spectrum values that lie between, and include, zero and the Nyquist frequency. In most conventional Fourier transform algorithms though, the output spectrum ranges between the negative and positive Nyquist frequencies. The negative frequency components are not unique from those at the positive frequencies, but the negative components are required to conserve the energy of the input signal, (Elliot and Rao, 1982).

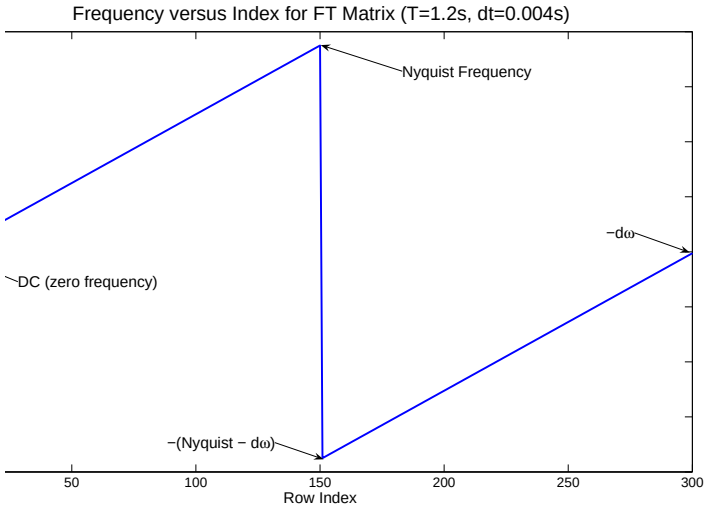

FIG. 3. Frequency axis for the forward Fourier transform matrix. The value of ω_j is plotted against the row index, *j*. The frequency axis for the inverse Fourier transform is the same, except it will vary with the column index. The frequency axis should be constructed to match a standard FFT algorithm - in this case, the goal was to match the FFT frequency axis used by MATLAB.

Realizing that there is more than one way to construct a frequency axis is important in order to construct a matrix operator to perform the Fourier transform. Since it makes sense to use a standard fast Fourier transform (FFT) algorithm whenever possible, it is often best to choose an axis that agrees with that of the FFT algorithm. For this work, we use the MATLAB FFT, which outputs a vector with the DC (or zero frequency) component in the first element, the positive and then negative Nyquist frequency components in the middle elements, and then the negative frequency components, decreasing in magnitude, in the subsequent elements. Even though frequency components are sampled at frequencies above the Nyquist frequency, they are just an aliased version of some other frequency component below the Nyquist frequency. The plot in Figure 3 displays the value of the frequency axis for a 1.2 second trace sampled at 4 ms. Also, to conform with the MATLAB FFT structure, we choose a frequency sampling interval, $d\omega = 2\pi/T$, where T is the maximum input axis value. For the example in Figure 3, T=1.2 seconds, and $d\omega$ = 5.236 rad/sec.

Fortunately, the frequency axis is the only part of constructing a Fourier transform matrix operator that is not straightforward. In equation 17, ω_j is the *j-th* element of the frequency axis, and t_l is the *l-th* element of the time axis. Since MATLAB starts indexing at one, and recording time usually starts at zero, $t_l = (l-1)dt$. Once the time and frequency axes are constructed, calculating the *(j,l)-th* element of the Fourier transform matrix is done with equation 17. Figure 4 displays the real part of the forward Fourier transform matrix for an input time series of length N=300, and dt=4ms.

The inverse Fourier transform can also be cast as a matrix operator. The only major difference is that the axes have interchanged, that is, now the row index corresponds to input frequency instead of input time, and the column index corresponds to output time instead of output frequency. Each element is calculated as an exponential with a positive sign now, and there is a scaling factor of $1/2\pi$. Other than the scaling factor, the inverse

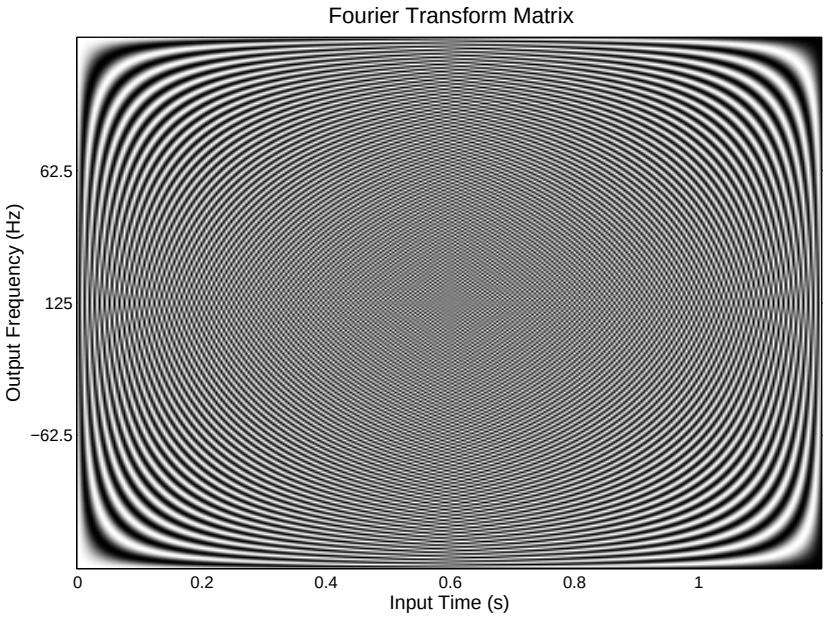

FIG. 4. The real part of a 300*x*300 Fourier transform operator matrix. Multiplying the full complex form of this matrix by a discrete time series yields the frequency spectrum of that time series.

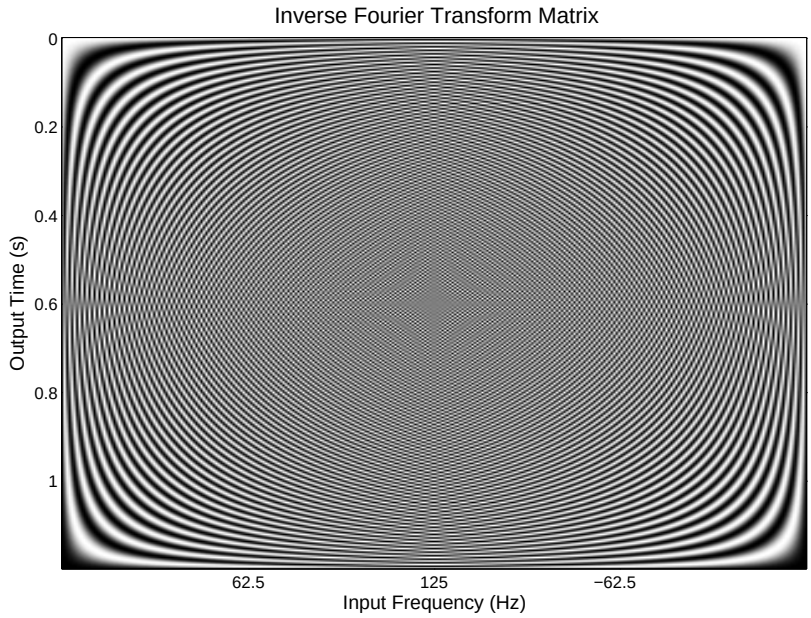

FIG. 5. The real part of a 300*x*300 inverse Fourier transform operator matrix. Multiplying the full complex form of this matrix by a discrete frequency spectrum recovers the time series associated with that spectrum.

Fourier transform matrix is the adjoint of the forward Fourier transform matrix, and the operation can be written as:

$$f(t_j) = \sum_{l=1}^{N} \left(\frac{e^{\mathrm{i}\omega_l t_j}}{2\pi} \right) F(\omega_l), \tag{18}$$

with no summation inside the large parentheses. The inverse Fourier transform matrix is shown in Figure 5.

Constructing a matrix operator for the forward and inverse data processing transforms follows the same procedure as we used for the Fourier transform. The general procedure is summarized here:

1. Identify the kernel of the transform and the input and output signals.
2. Construct the discrete input and output axes based on the sampling interval and domain of the input signal.
3. For an input signal of *N*-samples, initialize an empty matrix of size *NxN*, and associate the matrix indices with the input and output axes.
4. For each *(j,l)* matrix coordinate pair, calculate the kernel value and place it in the matrix.

The forward and inverse Fourier transform is often discussed as a mixed-domain transform between time and frequency domains. However, the Fourier and inverse Fourier transform are more generally described as a map between an input domain and its associated Fourier domain. In the same sense as the inverse Fourier transform, the forward data processing transform has an input Fourier-domain axis and an output axis that is associated with the original input domain. The only difference is that the input domain axis becomes distorted during the transform as a function of the regular input axis. The concise form of the forward data processing transform can then be cast as a matrix vector multiplication:

$$h(q_j) = \sum_{l=1}^{N} \frac{e^{\mathrm{i}\beta_l p(q_j)}}{2\pi} F(\beta_l). \tag{19}$$

If the input signal is a time series, as in an NMO correction, then p and q are time variables, and they would correspond to the vertical output axis of the NMO matrix. The input axis would correspond to β, and in the case of NMO, would be the frequency axis that is build to match the Fourier transform frequency axis. A convenient feature of the data processing transform matrices is that they do not differ from the form of the Fourier transform matrices other than the time axis, and scaling factors. So the standard Fourier frequency axis that is carefully matched to the FFT axis is still valid. The standard time

axis is still useful, as it actually represents q for a given index. A new time axis is needed with elements p_j, which are each just the value of $p(q_j)$ for the *j-th* row. If the input series to the data processing transform is a frequency spectrum, as in *f-k* migration, the axes take on different meanings.

The inverse data processing transform is more similar in form to the forward Fourier transform, in that it usually takes the series from the input domain and outputs a Fourier spectrum. Again following the procedure to construct a transform matrix, cast the inverse data processing transform (14) as a matrix-vector multiplication:

$$F(\beta_l) = \sum_{l=1}^{N} \frac{\left[\alpha_l e^{-\mathrm{i}\beta_j p(q_l)}\right]}{2\pi} f(p_l), \tag{20}$$

with no summation over *l* inside brackets. A standard inverse Fourier transform such as the inverse fast Fourier transform, (IFFT) can be applied to the output vector in equation 20 to recover the uncorrected input signal.

Although not necessary for implementation, it is insightful to separate the data processing transform matrices as a composition of a Fourier transform matrix with a shifting matrix. For a set of traces that are of the same length, the only part of the data processing transform matrix that changes from trace to trace is the shifting exponential. It is only because of the unique form of the shift itself that the exponentials reduce to a single transform exponential. Before this reduction, the Fourier transform matrices can be constructed separately from the forward and reverse shifting matrices for the same data. The construction of the Fourier transform matrices is already given above, and the forward shift matrix is given by:

$$a_{\mathrm{jl}} \equiv e^{\mathrm{i}\omega_l \Delta_j} = e^{\mathrm{i}\omega_l (p(q_j) - q_j)}. \tag{21}$$

The reverse shifting matrix is then given by:

$$b_{\mathrm{jl}} \equiv \frac{e^{-\mathrm{i}\omega_j \Delta_l}}{\alpha_l} = \frac{e^{-\mathrm{i}\omega_j (p(q_l) - q_l)}}{\alpha_l}, \tag{22}$$

with no summation over indices. This approach only requires the Fourier matrices to be calculated once for an input dataset. For that given dataset, the traces are looped over, and at each trace, the shifting matrix is computed, and then combined with the Fourier matrix by Hadamard (entry-wise) matrix product. This builds the transform matrix, which is the multiplied by either the corrected input trace for the inverse transform, or the spectrum of the uncorrected input trace for the forward transform.

The inverse of the data processing transform can be exactly formulated as seen in the continuous case. This alone is a significant development, but when cast as a matrix, the data processing transform has a form very similar to the Fourier matrices, and not only

that, its approximate inverse is easily calculated. From equations 21 and 22, we claim that the inverse matrix for each trace is best approximated by the adjoint of the forward matrix, scaled by α.

DISCUSSION

Casting a theoretically continuous input signal as a vector is a natural and correct way to view recorded seismic data. The time-sampling interval of the recorded data determines the Nyquist frequency of the recorded signal, (Gubbins, 2004), and it can be assumed that the data is frequency band-limited between zero and the positive Nyquist frequency, (Gubbins, 2004). This means that the recorded signal is exactly determined by this small and discrete range of frequencies. No information is gained by using filters that operate on frequencies higher than the Nyquist frequency. This observation, combined with the Fourier sampling theorem, suggests that a continuous, but band-limited form of the recorded signal exists, and it is exactly described by the Fourier basis. This continuous form is not necessarily the same as the true continuous function that a seismometer attempts to record. Conventional methods estimate the values of this continuous function by assuming that the function behaves like some interpolant between samples. We claim that a repeatable and discretely accurate approach to data processing is to return the exact value of the band-limited continuous equivalent of the recorded signal at any desired input coordinate. The data processing transform attempts to do exactly this.

The data processing transform is a special case of the nonstationary Fourier shift theorem. It follows then, for the continuous case, that data processing steps implemented by transform are effectively performing convolution with a shifting scaled Dirac delta function. The sifting property of the delta function justifies using convolution with a delta function to return the exact value, as only the value of the input signal at the target time is extracted. This is the ideal way to apply data processing corrections in both transform and time mapping approaches, but the discrete nature of seismic data prevents this. We suggest below that any interpolation scheme can be viewed as an approximation to a continuous and exact nonstationary shift. This suggestion arises from the observation that interpolation operators are approximations to the Dirac delta function.

The Dirac delta function can be formulated as the generalized limit of other more common functions, (Papoulis, 1962). Two clear examples of this are boxcar (rectangle) and Gaussian functions, which both approach an impulse as their widths go to zero and their heights go to infinity, (Papoulis, 1962). Take for example the boxcar function:

$$r_\epsilon(t) = \begin{cases} \frac{1}{\epsilon} & |t| \leq \epsilon \\ 0 & |t| > \epsilon \end{cases} . \tag{23}$$

As ϵ approaches zero, the width of the boxcar goes to zero, and the height goes to infinity, see figure 6. As this occurs, the boxcar function approaches the Dirac delta function:

$$\lim_{\epsilon \to 0} r_\epsilon(t) = \delta(t). \tag{24}$$

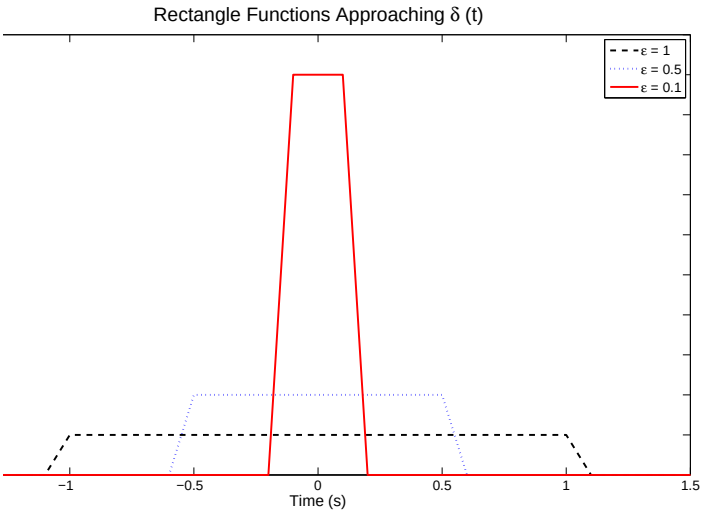

FIG. 6. Nearest-neighbour interpolant approaching $\delta(t)$. As ϵ approaches zero, the rectangle function defined by equation 23 approaches the Dirac delta function.

The scaled boxcar function is used as an interpolant for nearest-neighbour interpolation, where the width of the boxcar is equal to the sampling interval. This is a poor choice in most cases, as nearest-neighbour interpolation in the time domain has the unwanted effect of multiplying the frequency spectrum by a sinc function. Conversely, nearest neighbour interpolation in the frequency domain introduces ringing into the time series.

Linear interpolation is equivalent to convolution with a scaled triangle function with a width equal to twice the sampling interval, (Harlan, 1982). The triangle function also approaches the delta function as its width goes to zero and its height goes to infinity:

$$\Lambda_\epsilon(t) = \begin{cases} \frac{1}{\epsilon} - \frac{|t|}{\epsilon^2} & |t| \leq \epsilon \\ 0 & |t| > \epsilon \end{cases}, \tag{25}$$

$$\lim_{\epsilon \to 0} \Lambda_\epsilon(t) = \delta(t). \tag{26}$$

This behaviour can be seen in figure 7.

Data processing corrections implemented by sinc-interpolation algorithms effectively perform convolution of the input signal with a shifting scaled sinc-function. In the general continuous case, this has no clear justification, but for band-limited or discrete data, the purpose of the sinc-function becomes apparent, (Harlan, 1982). The Fourier transform of the sinc-function is a boxcar function over its own frequency content, (Harlan, 1982). Since the ideal interpolant should not affect the frequency content of the data while it estimates an intermediate value, a sinc function with frequency content at least up to the Nyquist frequency of the input signal will be the ideal interpolator, (Harlan, 1982). To reduce the Gibbs phenomenon of ringing associated with the transform of a boxcar function, tapering of the sinc function is often performed, (Rosenbaum and Boudeaux, 1981), and computa-

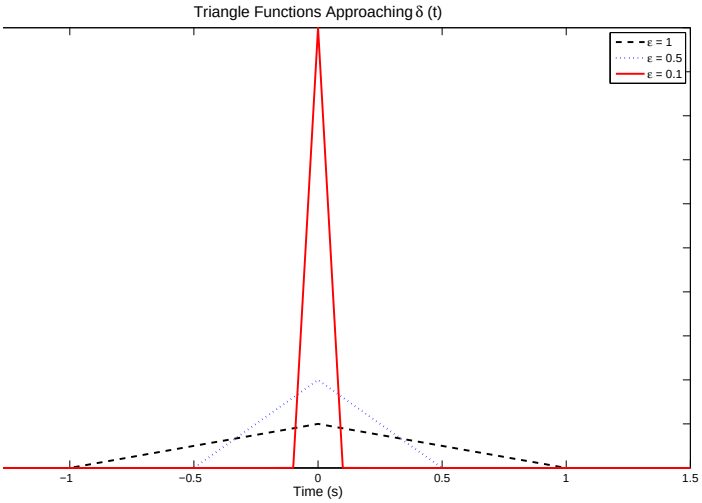

FIG. 7. Linear interpolant approaching $\delta(t)$. As ϵ approaches zero, the triangle function defined by equation 25 approaches the Dirac delta function.

tional efficiency is increased by truncating the length of the sinc operator, (Harlan, 1982).

The sinc function is a very good choice for an interpolant as it attempts to respect the finite bandwidth of the recorded data. However, convolution with a sinc function is still an approximation to convolution with the ideal delta function. For the following expression, (Elliot and Rao, 1982), as ϵ approaches infinity, the normalized sinc function approaches an impulse, see figure 8:

$$\mathrm{sinc}_\epsilon(t) = \frac{\epsilon \sin(\pi \epsilon t)}{\pi \epsilon t}, \tag{27}$$

$$\lim_{\epsilon \to \infty} \mathrm{sinc}_\epsilon(t) = \delta(t). \tag{28}$$

There are more similarities between the narrowing sinc function and the Dirac delta function than just the shape of the pulse. When ϵ is less than infinity in equation 28, the side lobes of the sinc function mimic GibbÂŠs phenomenon in Fourier transform theory, (Papoulis, 1962). Since the Fourier transform of a sinc function is a boxcar function, one can also visualize how it approaches the delta function in the Fourier domain. As the sinc function narrows, its frequency content increases, and its boxcar Fourier transform broadens. When the sinc function finally becomes an impulse, its Fourier spectrum is an infinitely wide boxcar, which is simply a constant, just as the Fourier transform of the delta function. Applying a band-pass filter to a delta function yields a sinc function, and conversely, the sinc-function is a band-limited estimation of the Dirac delta function. The sinc-function interpolant, *before* tapering or truncation, respects the Fourier basis assumed by the Fourier transform, and is equivalent to processing by discrete nonstationary filtering.

The inverse to any processing step is exactly formulated here for the continuous case

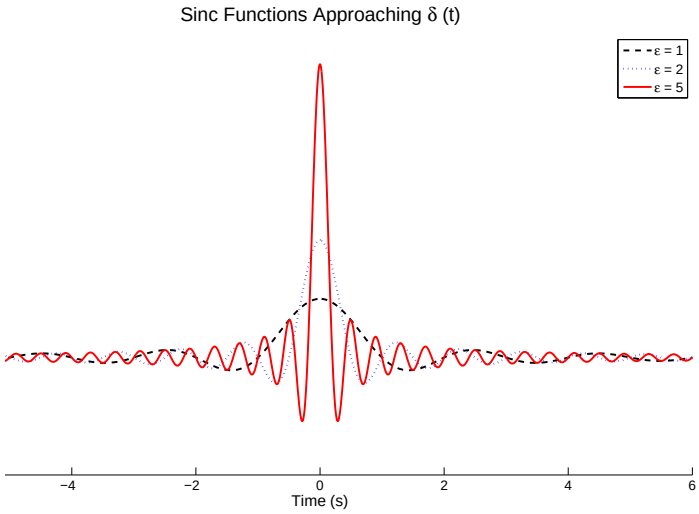

FIG. 8. Sinc interpolant approaching $\delta(t)$. As ϵ approaches infinity, the sinc-function defined by equation 27 approaches the Dirac delta function.

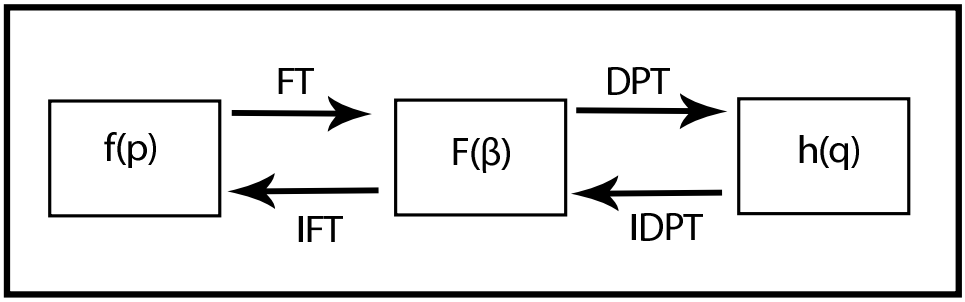

FIG. 9. Mixed-domain transform diagram. The relation between the Fourier transform (FT), the data processing transform (DPT), the inverse Fourier transform (IFT), and the inverse data processing transform (IDPT) is shown here. The FT and IFT move between the uncorrected and Fourier domains, while the DPT and IDPT move data between the Fourier and corrected domains.

using the nonstationary version of the Fourier shift theorem. This is not surprising, as even the conventional procedure of mapping data directly between input and output times is exactly reversible assuming continuous data. Casting the integral kernel as a discrete matrix defined on the sampling intervals allows a precise method of approximating the exact continuous form assuming a Fourier basis. Although the continuous inverse of a nonstationary shift is exact, the matrix approximation can be evaluated by checking if the product of the forward and inverse transform matrices for a given trace yields an identity matrix.

Like the Fourier transform, the data processing transform is a mixed-domain transform. In the general case, the input data should be viewed as an image, and then a distorted version of that image is the desired output. The data processing transform moves data directly between the Fourier spectrum of the input image and the corrected image (See figure 9). Viewing the input and output data each as an image makes the general application of the transform to any processing step clear. The physical meaning of the transform coordinates and distortions can be defined only when discussing specific processing steps.

ACKNOWLEDGEMENTS

The Jackson School of Geosciences, University of Texas at Austin, the EDGER Forum, and the sponsors of CREWES for supporting this research.

REFERENCES

Claerbout, J. F., 1992, Earth Soundings Analysis: Processing versus Inversion: Blackwell Science.

Elliot, D. F., and Rao, K. R., 1982, Fast Transforms: Algorithms, Analysis, Applications: Academic Press, Inc.

Gubbins, D., 2004, Time Series Analysis and Inverse Theory for Geophysicists: Cambridge University Press.

Harlan, W., 1982, Avoiding interpolation artifacts in stolt migration: SEP, **30**, 103–110.

Karl, J. H., 1989, An Introduction to Digital Signal Processing: Academic Press.

Margrave, G. F., 1998, Theory of nonstationary linear filtering in the fourier domain with application to time-variant filtering: Geophysics, **63**, 244–259.

Margrave, G. F., 2001, Direct fourier migration for vertical velocity variations: Geophysics, **66**, 1504–1514.

Margrave, G. F., and Ferguson, R. J., 1999, Wavefield extrapolation by nonstationary phase shift: Geophysics, **64**, 1067–1078.

Papoulis, A., 1962, The Fourier Integral and its Applications: McGraw Hill Classic Textbook Reissue Series.

Rosenbaum, J. H., and Boudeaux, G. F., 1981, Rapid convergence of some seismic processing algorithms: Geophysics, **46**, 1667–1672.

Sneddon, I. N., 1995, Fourier Transforms: Dover Publications.

Yilmaz, Ť., 2001, Seismic Data Analysis: Society of Exploration Geophysics.