Stanford Exploration Project, Report 77, November 17, 1997, pages 301–316

Integration along a line in a sampled space

Dave Nichols[1]

keywords: *interpolation, approximation, linear*

ABSTRACT

Integration along a line in a sampled space can be thought of as a two stage process: first interpolate to a continuous space and then perform the integral. To compute a line integral in seismic data I first apply a weighting funtion to each trace and then stack the traces. The form of the weighting function depends on the interpolation method chosen.

INTRODUCTION

Many geophysical algorithms can be cast in the form of a line integral in the time-space domain. The most commonly used are NMO-Stack, DMO , slant stack and Kirchhoff migration. The integral may be written in several ways, as an integral over angle, over slowness, over space, over time, or over arc length. Since seismic data is collected at discrete spatial locations, these integrals are most often implemented as a space integral (although this is not the case when the time coordinate of the curve is a multi-valued function of space). The integrals are derived in a continuous space but when they are applied to seismic data they are implemented in a discretely sampled space.

There are many ways to approximate the continuous integral. One of the most common is to sample the curve evenly in space and sum a single sample for each spatial location. Hale (1991) demonstrated that this simple approach can produce operator aliasing when the operator has steep dips. He suggested the use of equal time sampling to overcome this problem. However, equal time sampling will not sample all the available information when the dips are flat.

I propose a simple way of looking at different strategies for performing the integration and their use on a seismic dataset. The line integral should be just that, an integral along the line! The problem then becomes one of interpolating the sampled data from its discrete locations to a continuous function along the line, and integrating

[1]**email:** not available

the result. All of the interpolation/integration strategies can be coded as a weighted summation of one or more time samples at each trace location. The anti-aliasing Kirchhoff migration proposed by Claerbout (Claerbout, 1992) is also in the form of a weighted summation of samples on each trace. I discuss the relationship between his methods and the methods I propose here.

A MODEL PROBLEM

To illustrate the various interpolation techniques I will use a 2-D surface that is a plane wave of frequency, ω, and slowness, p,

$$f(x,t) = sin(\omega(t-px)).$$

This function is shown in Figure 1 for a frequency of 20Hz and a slowness of 1/2000 ms^{-1}.

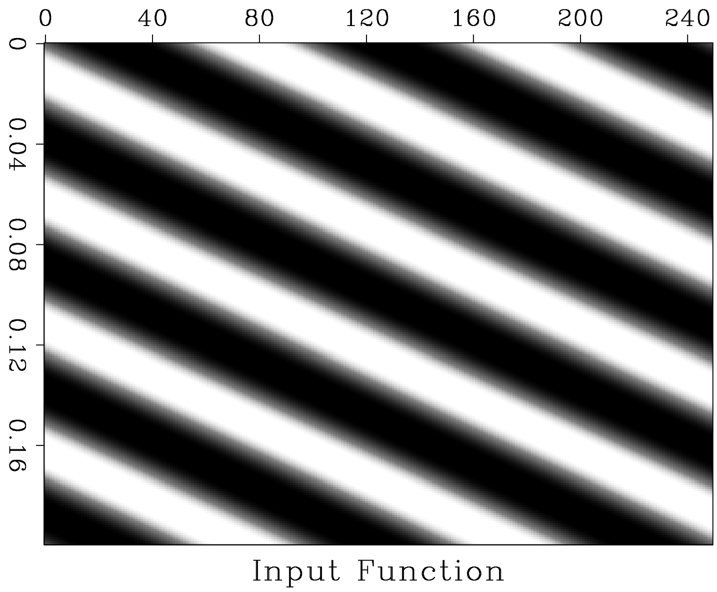

Figure 1: Example continuous data, a mono-frequency plane wave with frequency 20Hz and slowness 1/2000 ms^{-1}. dave1-raw-func [ER]

If this data is sampled using typical seismic data sampling rates, $4ms$ in time and $25m$ in space, the data shown in figure 2 is obtained. This is the input data to the various interpolation schemes,

$$\tilde{f}(ix,it) = f(x_{ix},t_{it})\,;\ ix \in \{1,\ldots,nx\}, it \in \{1,\ldots,nt\}$$

The data is sampled at coordinates given by x_{ix} and t_{it}.

In the following examples I consider the case of an integral path on a 2-D surface that is defined by time as a function of space, $t(x)$. The aim of this paper is to obtain

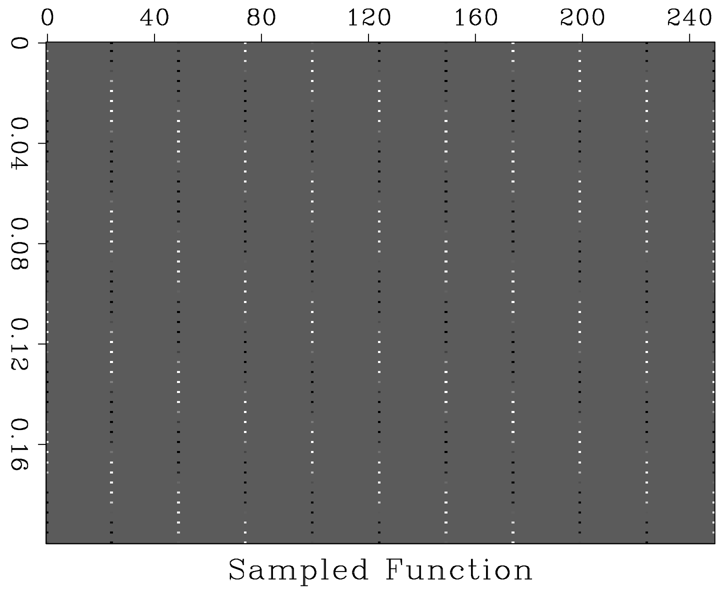

Figure 2: Input data sampled every $4ms$ in time and $25m$ in space. dave1-sampled-func [ER]

the integral along a particular path as the sum over all the traces of the trace scaled by a weighting function which is localized around the intersection of the integration trajectory and each trace, $t(x_{ix})$, with nearest sample point $it(ix)$. The curve integral can then be expressed as a sum over the sampled data points.

$$\int_{xmin}^{xmax} f(x, t(x))\, dx = \sum_{ix=1}^{nx} \sum_{j=-n}^{+m} w(ix, it, j) \tilde{f}(ix, it(ix) + j)$$

INTERPOLATION STRATEGIES

Sampling in space

This is the simplest algorithm and one that is frequently used. At each trace location the time at which the curve crosses the trace is calculated. A value for the trace at this time is then added into the integral. The value may be the value at the nearest time sample, or it may be a value interpolated from the neighboring time samples.

This method is illustrated in figure 3. The sample value at the trace crossing has been assigned to the region of the curve between $(x_{ix} + x_{ix-1})/2$ and $(x_{ix+1} + x_{ix})/2$. Each trace is sampled at a single location. This approach can lead to operator aliasing when the dip of the summation trajectory is steep. The plot on the right of figure 3 shows a weighting function for each trace, the integration can be implemented by multiplying each trace by the weighting function and summing. The weighting

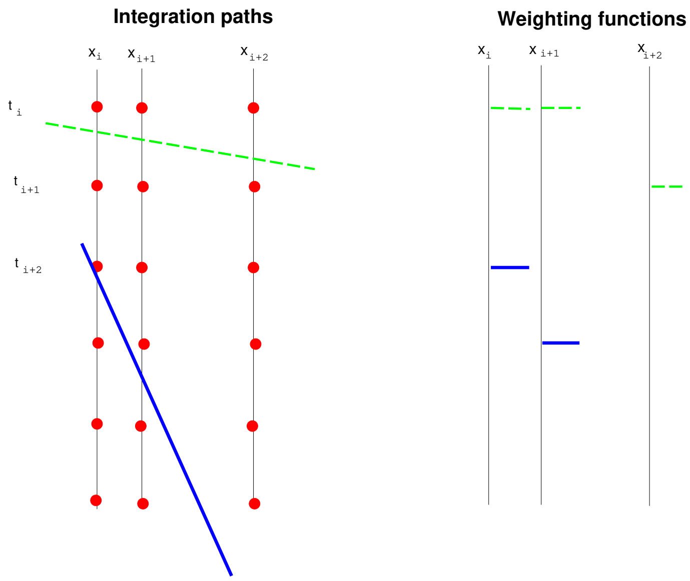

Figure 3: Sampling in space, a single sample is used at each space location. The weighting function is a set of impulses, one on each trace. Two trajectories and their corresponding weighting functions are shown. The upper, dashed, trajectory is relatively flat; the lower, solid trajectory is steep. $\boxed{\text{dave1-samp}}$ [NR]

function depends on the sampling between the traces. Since this is a space integral the weight at the sampled point will be the average distance to the neighboring traces, i.e. the spatial length of the integration path for which the sample value is used.

Sampling in time

Hale's method assumes that the data is adequately sampled in time but that problems will occur if the integral is merely sampled in space. He therefore uses an integral equally sampled in time and uses either sinc or nearest neighbor interpolation in space to perform the integration. He illustrates that nearest neighbor interpolation can be sufficient for a typical seismic dataset. However this method does not use all the available data when the integration path is flat; the equal time sampling generates widely spaced spatial samples.

Continuous in time, nearest neighbor in space

Before considering interpolation in both space and time I will first consider two cases where only interpolation in space is required. In these methods the traces are assumed to be continuous functions of time.

$$\tilde{f}(ix,t) = f(x_{ix},t) \;;\; x \in \{1,\ldots,nx\}$$

The first method uses nearest neighbor interpolation in space. The value at every point on the curve is taken from the nearest trace.Figure 4 shows the values interpolated at every point on the trajectory. As the trajectory gets steeper a greater length of each trace is used in the integral. The trace is used from the time at which the trajectory crosses $(x_{ix} + x_{ix-1})/2$ to the time at which it crosses $(x_{ix+1} + x_{ix})/2$. The function $\hat{f}(x,t)$ is the function through which the line integral is actually calculated. If this is a good approximation to the true function the the integral will be close to the true integral.

$$\hat{f}(x,t) = \tilde{f}(ix,t), \ (x_{ix} + x_{ix-1})/2 < x < (x_{ix+1} + x_{ix})/2$$

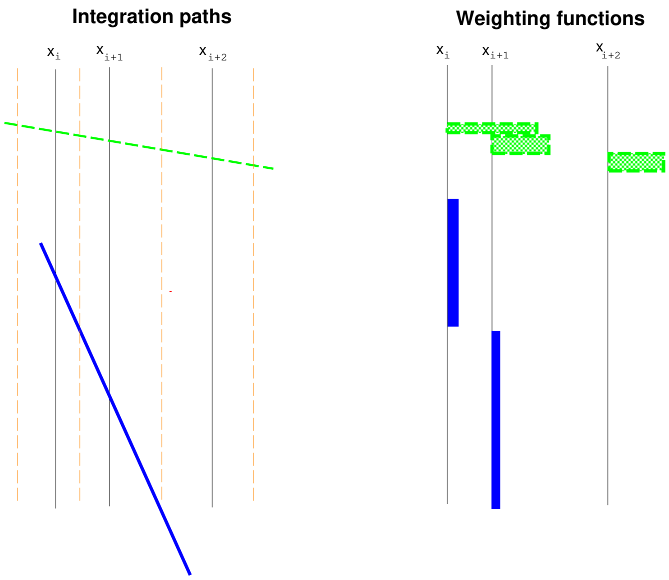

Figure 4: Continuous in time, nearest neighbor in space. The value at every point on the path is taken from the nearest trace. The weighting function is a rectangle on each trace. dave1-ct-nns [NR]

The weighting function on each trace is a rectangle of area equal to the average distance to the neighbor traces, note that this is because our integral is cast as an integral in x; if it were an integral in t, the rectangles would all be of unit height.

This method is closely related to the first method described by Claerbout, (1992) where a rectangular weights are used to perform an anti-aliased Kirchoff migration.

Continuous in time, linear interpolation in space

Instead of using nearest neighbor interpolation in space I now use linear interpolation in space. The value at every point on the path is linearly interpolated from the two neighboring traces.

Figure 5 shows the region of the traces used in this method. The trace is used from the point at which the trajectory crosses the previous trace to that at which it

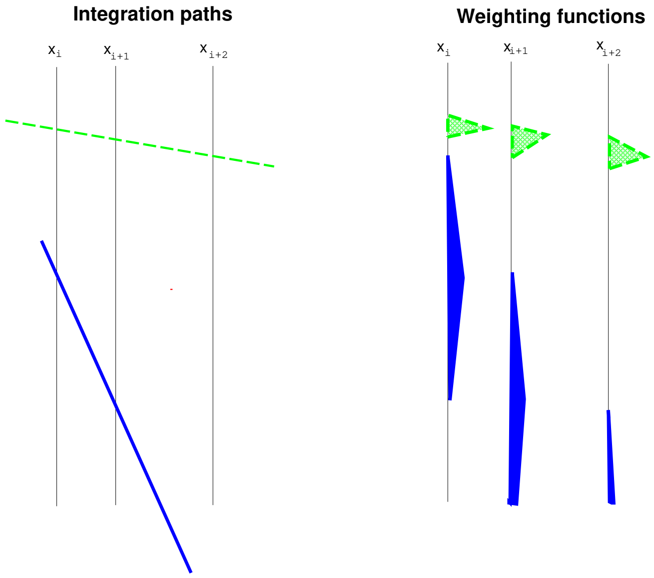

Figure 5: Continuous in time, linear interpolation in space. The value at every point on the path is linearly interpolated from the two surrounding traces. The weighting function is a triangle on each trace. dave1-ct-lins [NR]

crosses the next trace. At each point on the curve the value is linearly interpolated from the two surrounding traces. So that

$$\hat{f}(x,t) = \tilde{f}(ix,t)(1. - l_{ix}) + \tilde{f}(ix+1,t)l_{ix}$$

where, $l_{ix} = (x - x_{ix})/(x_{ix+1} - x_{ix})$.

The weighting function is a triangle. It has its peak at the point that the trajectory crosses the trace, and it has an area equal to the average distance to the neighbor traces. If the integral were a time integral, the limits of the triangle would be the same but it would have a unit peak height.

This method is closely related to the second method described by Claerbout, in which he uses triangular weights in Kirchoff migration.

Nearest neighbor in time and space

In this method I account for the fact that both time and space are discretely sampled. Figure 6 shows the region subdivided into cells. Within each cell the function value is approximated by the value closest to the center of the cell.

The integral within a cell for this approximation is given by,

$$\int_{\xi_0}^{\xi_1} \tilde{f}(ix,it)dx = \tilde{f}(ix,it)(\xi_1 - \xi_0)$$

where ξ_0 and ξ_1 are the space coordinates at which the path enters and leaves the cell. The integral along the whole path is the summation of a set of constant values weighted by the width of the path that lies within each cell (for a space integral.)

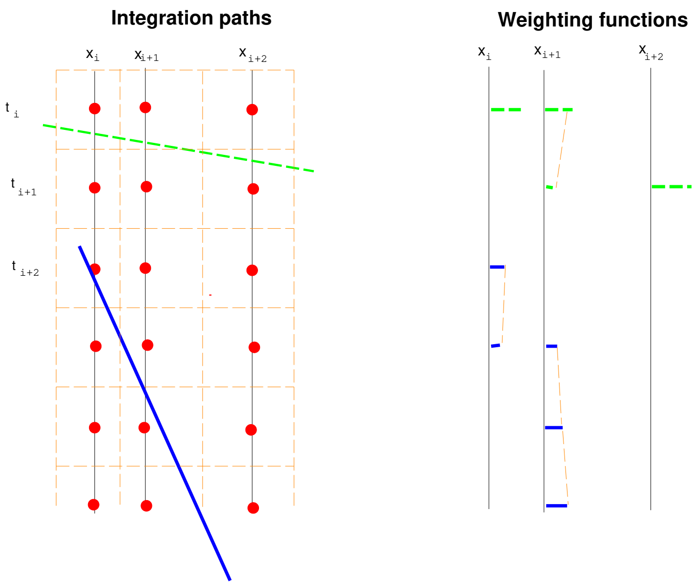

Figure 6: Nearest neighbor is space and time. The function is approximated by a constant value in each cell. The weighting function is the path width within the cell. dave1-nnt-nns [NR]

A cell is bounded by planes midway between the sample points. There are three types of path through a cell.

1. When the integration path is near horizontal, it will lie completely within one cell, the sample within that cell will then be scaled by the width of the cell.
2. If the path passes completely through a cell from top to bottom, the sample will be weighted by the width of the path within that cell. This width can be calculated from the slope of the path.
3. There are also cases where the path passes through the corner of a cell. In this case the width of the path within the cell is a function of the point at which it enters the cell and the slope.

These three cases have to be dealt with separately when this method is implemented. Note that because of the corner cases this method is not a rectangular weight function on each sampled trace. The method of rectangles proposed by Claerbout implicitly assumes that the path either passes completely through a cell or misses it altogether.

Figure 7 shows the approximation to the original function surface that is implied by this method. It is clear that for some integration paths the integral through this surface will be a poor approximation to the true integral.

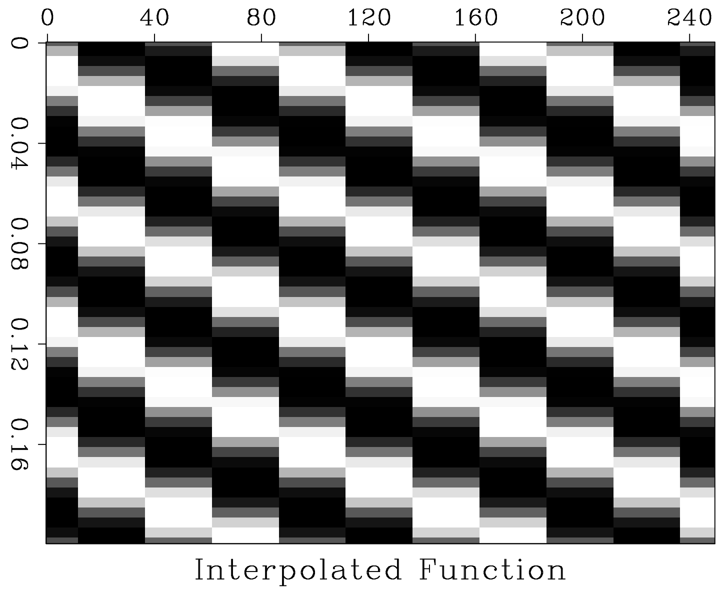

Figure 7: Input data sampled every 4ms in time and 25m in space and then interpolated using nearest neighbor interpolation in space and time. dave1-nnxnnt-func [ER]

Nearest neighbor in time, linear interpolation in space

If the function is approximated by linear interpolation in space then the integral has a different form. A single sample will influence the interpolated value within two regions. Each region is bounded by a neighboring trace and half the distance to the nearest time samples.

Within each regions the approximate function is,

$$\hat{f}(x,t) = \tilde{f}(ix-1,it)\left[1 - \frac{x - x_{ix-1}}{x_{ix} - x_{ix-1}}\right] + \tilde{f}(ix,it)\left[\frac{x - x_{ix-1}}{x_{ix} - x_{ix-1}}\right]$$

and the integral of the approximate function is given by,

$$\int_{\xi_0}^{\xi_1} \tilde{f}(ix-1,it)\left[1 - \frac{x - x_{ix-1}}{x_{ix} - x_{ix-1}}\right] + \tilde{f}(ix,it)\left[\frac{x - x_{ix-1}}{x_{ix} - x_{ix-1}}\right] dx$$

This has two terms, one involving each sample value.

$$\frac{(\xi_1 - \xi_0)}{(x_{ix-1} - x_{ix})}\left[\tilde{f}(ix-1,it)(\frac{\xi_0 + \xi_1}{2} - x_{ix}) \;+\; \tilde{f}(ix,it)(x_{ix-1} - \frac{\xi_0 + \xi_1}{2})\right]$$

The weight for one sample point is the sum of the weights from the cell to its left and right. When calculating the integral along a path the paths fall into the same three classes as in the previous method. The entry and exit points of the path within a cell must be calculated, and the weights for the two samples that define the cell can then be applied. Figure 8 illustrates the weights for this method. Although this method is close to the triangle weighting method proposed by Claerbout, the weights in a sampled space are not exactly triangles.

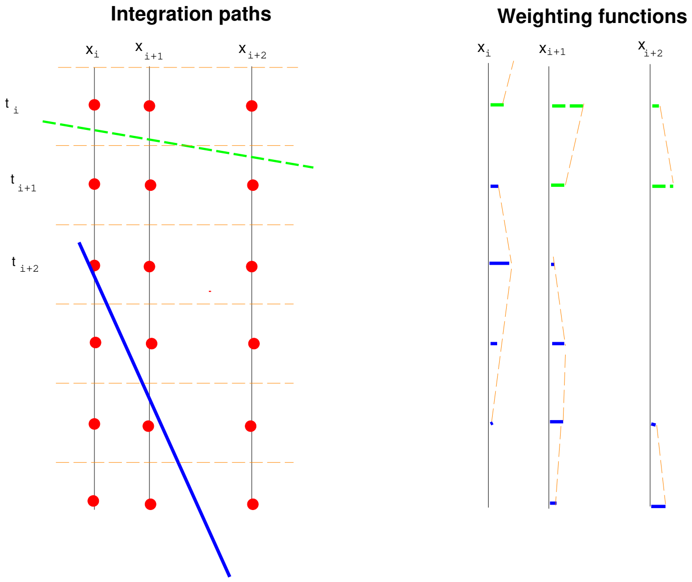

Figure 8: Nearest neighbor in time, linear interpolation in space. The function is approximated by linear interpolation in space within each cell. dave1-nnt-lins [NR]

Figure 9 shows the approximation to the original function surface that is implied by this method. If the time sampling is sufficiently fine this method will give a reasonable approximation of the true surface. When the time sampling is larger some interpolation in time must be considered.

Bilinear interpolation in time and space

In the case of bilinear interpolation, the approximate function is given by,

$$
\begin{aligned}
\hat{f}(x,t) &= \tilde{f}(ix-1,it-1)\left[1-\frac{x-x_{ix-1}}{x_{ix}-x_{ix-1}}\right]\left[1-\frac{t-t_{it-1}}{t_{it}-tit-1}\right] \\
&+ \tilde{f}(ix-1,it)\left[1-\frac{x-x_{ix-1}}{x_{ix}-x_{ix-1}}\right]\left[\frac{t-t_{it-1}}{t_{it}-tit-1}\right] \\
&+ \tilde{f}(ix,it-1)\left[\frac{x-x_{ix-1}}{x_{ix}-x_{ix-1}}\right]\left[1-\frac{t-t_{it-1}}{t_{it}-tit-1}\right] \\
&+ \tilde{f}(ix,it)\left[\frac{x-x_{ix-1}}{x_{ix}-x_{ix-1}}\right]\left[\frac{t-t_{it-1}}{t_{it}-tit-1}\right]
\end{aligned}
$$

Figure 10 illustrates the geometry associated with bilinear interpolation within a cell.

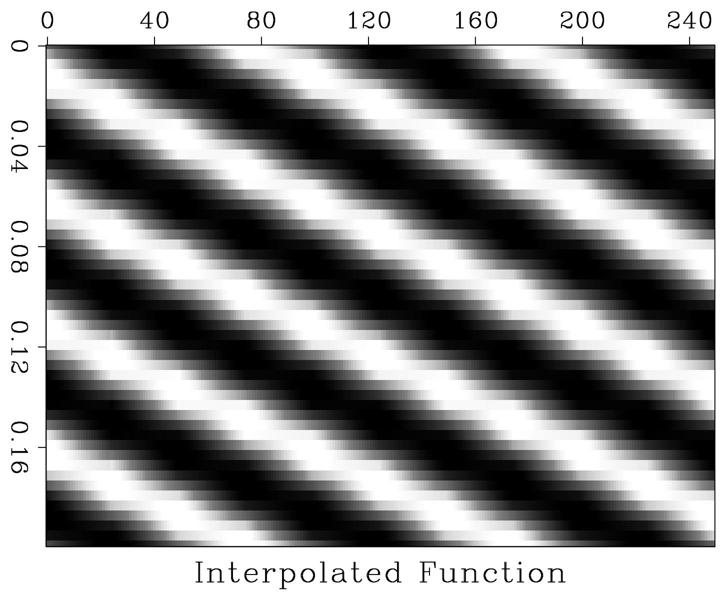

Figure 9: Input data sampled every 4ms in time and 25m in space and then interpolated using nearest neighbor in time and linear interpolation in space. dave1-linxnnt-func [ER]

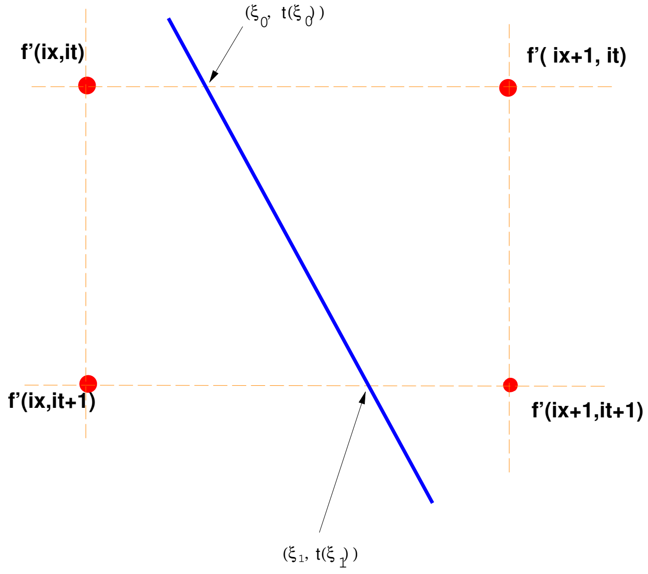

Figure 10: Bilinear interpolation. The function is approximated by bilinear interpolation within a cell defined by the four corner samples. The interpolation weights depend on the coordinates at which the path enters an leaves the cell. dave1-bilin [NR]

The path within the cell, $t(x)$, is approximated by a linear segment,

$$\hat{t}(x) = t(\xi_0)\left[1 - \frac{x - \xi_0}{\xi_1 - \xi_0}\right] + t(\xi_1)\left[\frac{x - \xi_0}{\xi_1 - \xi_0}\right]$$

The Mathematica code in the appendix will calculate the appropriate weights for each sample point. For example, the weight to be applied to the sample point (x_{ix}, t_{it}) is:

$$(\xi_1 - \xi_0)\frac{3(t_{it+1}\xi_0 + \tau_1 x_{ix+1} + t_{it+1}\xi_1 + \tau_0 x_{ix+1}) - 6t_{it+1}x_{ix+1} - 2(\tau_0\xi_0 + \tau_1\xi_1) - \tau_0\xi_1 - \tau_1\xi_0}{6(t_{it} - t_{it+1})(x_{ix+1} - x_{ix})}$$

Figure 11 shows the approximation to the original function surface that is implied by this method. For this set of parameters it is a reasonable approximation of the original surface. In general bilinear interpolation is a reasonable interpolation method for sampling intervals up to about half of the Nyquist frequency. If the data has high dips and high frequencies the approximation may not be valid.

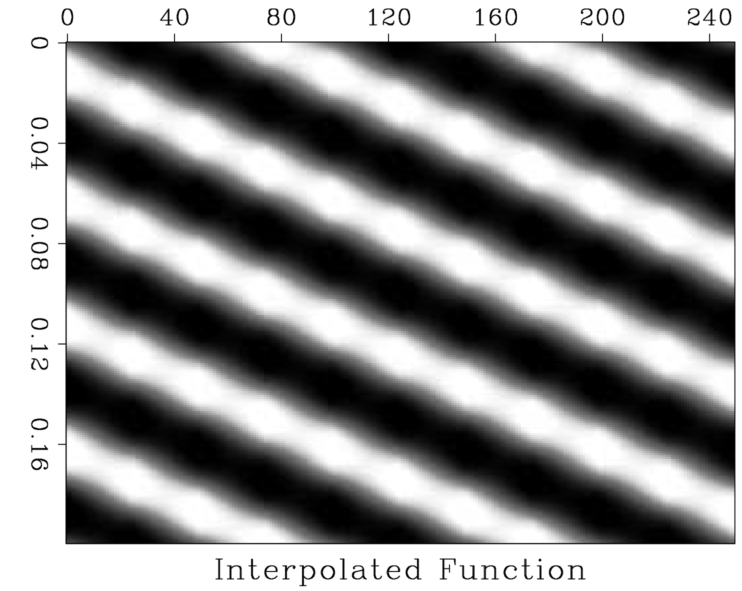

Figure 11: Input data sampled every 4ms in time and 25m in space and then interpolated using bilinear interpolation in space and time. dave1-bilin-func [ER]

DISCUSSION

I have shown how an integral along a path in a sampled space can be formulated as a weighted sum of the input samples. The form of the weighting function and the effort needed to calculate it depends on the way in which the input data is interpolated onto the path.

If an integral is approximated by taking equally spaced samples in space there may be problems associated with "operator aliasing". The nearest neighbor and/or linear interpolators that I have described can be used to reduce these problems. However these methods will not overcome problems related to aliasing in the sampled data. The assumption that a linear or bilinear interpolation of the data is a good approximation to the true data is violated in this case. Putting it another way, you can integrate along a path steeper than the aliasing limit if you use an interpolating integral, but this will not save you if the data itself is aliased at that dip.

The same method outlined here can be extended to treat a surface integral in a 3-D space, e.g. 3-D Kirchoff migration. It is left as an exercise for the reader.

REFERENCES

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

Hale, D., 1991, A nonaliased integral method for dip moveout: Geophysics, **56**, no. 6, 795–805.

APPENDIX

Mathematica code to calculate weighting functions for bilinear interpolation.

```
(* Bilinear approximating function within a cell bounded by x1,y1,x2,y2 *)
(* values at corners of cell are    *)
(*           (x1,y1) => f1      *)
(*           (x1,y2) => f2      *)
(*           (x2,y1) => f3      *)
(*           (x2,y2) => f4      *)
bilin[ x_, t_ ] := f1 ( 1- (x-x1)/(x2-x1))(1-(t-t1)/(t2-t1) ) + \
                   f2 ( 1 - (x-x1)/(x2-x1)) (t-t1)/(t2-t1) + \
                   f3 (x-x1)/(x2-x1) (1-(t-t1)/(t2-t1) ) +  \
                   f4  (x-x1)/(x2-x1) (t-t1)/(t2-t1)

(* A linear path across the cell *)
(* Starts at xx0,tx0, exits at xx1,tx1 *)
tau[ x_ ] := tx0 ( 1-(x-xx0)/(xx1-xx0)) + tx1 (x-xx0)/(xx1-xx0)

(* The bilinear function along the path *)
bilin[ x_ ] := bilin[ x, tau[ x ] ]

(* Integral along the path *)
res = Integrate [ bilin[x], {x, xx0, xx1 } ]

Print[""]
Print["Coefffficient in f1"]
cf1 = Simplify[ Coefficient[ Collect[ res, f1],f1]]
Print[""]
TeXForm[ cf1 ]
```

```
Print[""]
Print["Coeffficient in f2"]
cf2 = Simplify[ Coefficient[ Collect[ res, f2],f2]]
Print[""]
TeXForm[ cf2 ]

Print[""]
Print["Coeffficient in f3"]
cf3 = Simplify[ Coefficient[ Collect[ res, f3],f3]]
Print[""]
TeXForm[ cf3 ]

Print[""]
Print["Coeffficient in f4"]
cf4 = Simplify[ Coefficient[ Collect[ res, f4],f4]]
Print[""]
TeXForm[ cf4 ]
```