Stanford Exploration Project, Report 95, October 14, 1997, pages 1–14

Multidimensional recursive filters via a helix

Jon Claerbout[1]

keywords: *helix, imaging, signal-processing, recursion, Poisson*

ABSTRACT

A wire is coiled into a helix whose surface is a cylinder. I show that a filter on the 1-D space of the wire mimics a 2-D filter on the cylindrical surface. Thus 2-D convolution can be done with a 1-D convolution program. Likewise, I show some curious examples of 2-D recursive filtering (also called 2-D deconvolution or 2-D polynomial division). In 2-D as in 1-D, the computational advantage of recursive filters is the speed with which they propagate information over long distances. We can estimate 2-D prediction-error filters (PEFs), that are assured of being stable for 2-D recursion. Such 2-D and 3-D recursions vastly speed the solution of a wide class of geophysical estimation problems. Simple tasks that are vastly speeded are (1) estimating missing values on a multidimensional cartesian mesh, and (2) distributing irregularly positioned data on a regular mesh.

INTRODUCTION

For many years it has been true that our most powerful signal-analysis techniques are in *one*-dimensional space, while our most important applications are in *multi*-dimensional space. We have made a giant step towards overcoming this difficulty.

Recently, I noticed that a helix maps 2-D to 1-D and that convolutions and deconvolutions in one dimension are practically equivalent to their multidimensional forms. First I will explain the helical transformation. Then I will show a simple 1-D computer program and some 2-D convolutions and deconvolutions done with it.

Finally I will explain why there should be a big future for the idea of filtering in multidimensional space as though it were simply filtering on a line. The main benefit arises from recursive operations because they are very efficient at carrying information for long distances.

[1]**email:** jon@geo.stanford.edu. http://sepwww.stanford.edu/sep/jon/helix.ps This paper was presented at the August 3-7, 1997 summer research workshop of of the Society of Exploration Geophysics in Vail, CO. An earlier version of this paper was presented at the May 11-14, 1997 meeting of the sponsors of the Stanford Exploration Project and was included there as the last chapter of the GEM lectures.

CONVOLUTION ON A HELIX

Figure 1 shows some two-dimensional shapes that are convolved together. This convolution was done in one dimension, although the figure makes it appear that the convolution is two-dimensional.

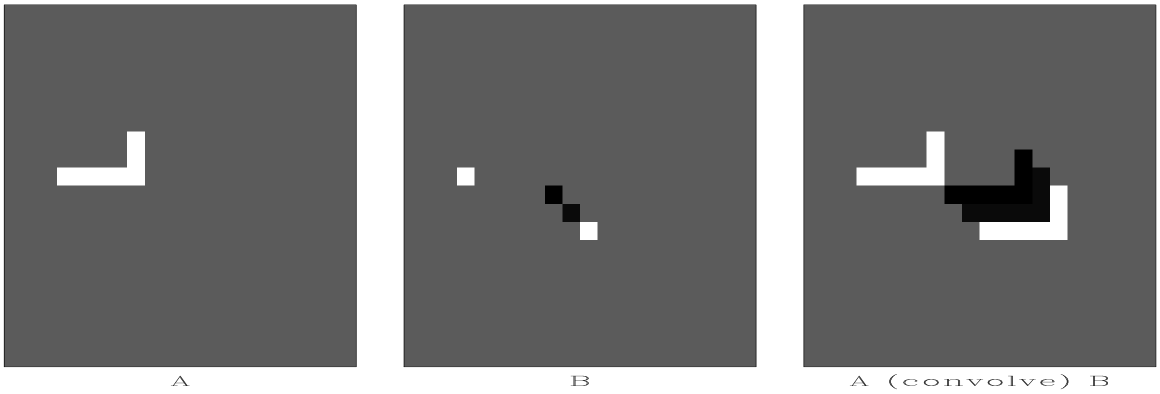

Figure 1: Two-dimensional convolution (as performed in one dimension). jon1-diamond [ER]

First we must map two dimensions into one. Wind a rope around your arm into a helical coil that neatly covers your arm. This coil is a two-dimensional surface while the rope is one-dimensional. Now paint a seismic signal on one of the loops around your arm, and likewise paint all the loops with other signals so you have a seismic section on your arm. It is OK if the backside of your arm has no paint on it; we'll call that zero padding.

Seismologists sometimes use the word "supertrace" to describe a collection of seismograms stored "end-to-end". Fortran programmers will recognize that Fortran's way of storing 2-D arrays in one-dimensional memory is exactly what we need for this helical mapping. Likewise, in 3-D we simply append one plane after another (like a 3-D Fortran array). It is easier to code than to explain or visualize a spool or torus wrapped with string, etc.

A basic idea of filtering, be it in one dimension, two dimensions, or more, is that you have some filter coefficients and some sampled data; you pass the filter over the data; at each location you find an output by crossmultiplying the filter coefficients times the underlying data and summing the terms. The order in which you produce the outputs is irrelevant. You could compute the results in parallel. Figure 2 shows a filter and some data organized on a helical coil. Since the output values can be computed in any order, we can slide the filter coil over the data coil in any direction. We could, however, slide the filter over the data in the screwing order that a nut passes over a bolt. The screw order is the same order that would be used if we were to unwind the coils into one-dimensional strips and convolve them across one another. The same filter coefficients overlay the same data values if the 2-D coils are unwound into 1-D strips. The helix idea allows us to obtain the same convolution output in either of two ways, a one-dimensional way, or a two-dimensional way. I used the one-dimensional way to compute the obviously two-dimensional result in Figure 1.

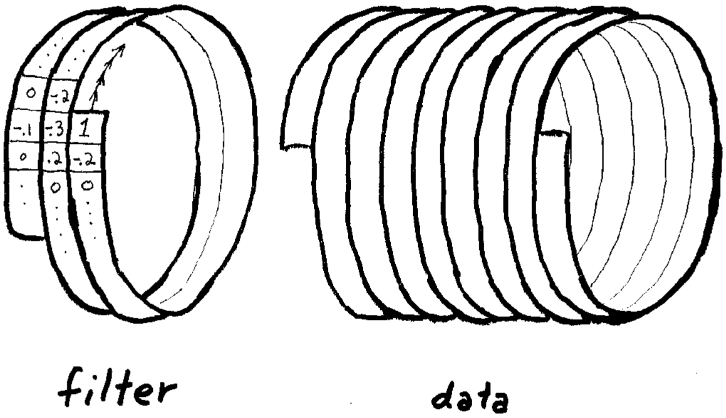

Figure 2: Filtering on a helix. The same filter coefficients overlay the same data values if the 2-D coils are unwound into 1-D strips. jon1-coil [NR]

Deconvolution (polynomial division) can undo convolution (polynomial multiplication). A magical property of the helix is that we can consider 1-D convolution to be the same as 2-D convolution. Hence is a second magical property: We can use 1-D *de*convolution to undo convolution, whether that convolution was 1-D or 2-D. Thus, we have discovered how to undo 2-D convolution. We have discovered that 2-D deconvolution on a helix is equivalent to 1-D deconvolution. We have learned how to do multidimensional deconvolution.

Deconvolution is recursive filtering. Recursive filter outputs cannot be computed in parallel, but must be computed sequentially as in one dimension, namely, in the order that the nut screws on the bolt.

Recursive filtering sometimes solves big problems with astonishing speed. It can propagate energy rapidly for long distances. Unfortunately, recursive filtering can also be unstable. There is a large literature and extensive technology about recursive filtering in one dimension. The helix allows us to apply that technology to two (and more) dimensions. It is a huge technological breakthrough.

BASIC REVIEW OF 1-D RECURSIVE FILTERS

Convolution is the operation we do on polynomial coefficients when we multiply polynomials. Deconvolution is likewise for polynomial division. Often these ideas are described as polynomials in the variable Z. Take $X(Z)$ to denote the polynomial whose coefficients are samples of input data, and let $A(Z)$ likewise denote the filter. The convention I adopt here is that the first coefficient of the filter has the value $+1$, so the filter's polynomial is $A(Z) = 1 + a_1 Z + a_2 Z^2 + \cdots$. To see how to convolve, we now identify the coefficient of Z^k in the product $Y(Z) = A(Z)X(Z)$. The usual case (k larger than the number N_a of filter coefficients) is

$$y_k \quad = \quad x_k + \sum_{i=1}^{N_a} a_i x_{k-i} \tag{1}$$

Deconvolution is a complicated process in seismology that includes estimating $A(Z)$. More simply, let us take the filter $A(Z)$ as known, the output $Y(Z)$ as known, and we want to go back to find the input $X(Z) = Y(Z)/A(Z)$. Again we simply identify the coefficient of each power of Z^k in $Y(Z) = A(Z)X(Z)$, but now we seek to recursively find x_k instead of y_k. Rearranging (1) we get

$$x_k \quad = \quad y_k - \sum_{i=1}^{N_a} a_i x_{k-i} \tag{2}$$

where now we are finding the output x_k from its past outputs x_{k-i} and from the present input y_k. We see that the deconvolution process is essentially the same as the convolution process, except that the filter coefficients are used with opposite polarity; and they are applied to the past *outputs* instead of the past *inputs*. That is why deconvolution must be done sequentially while convolution can be done in parallel.

EXAMPLES OF SIMPLE 2-D RECURSIVE FILTERS

Figures 3 and 4 contain this 2-D filter

$$\begin{bmatrix} 0 & -1/4 \\ 1 & -1/4 \\ -1/4 & -1/4 \end{bmatrix} \tag{3}$$

Let us experiment using this 2-D filter as a recursive filter. In Figure 3 the input is shown on the left. This input contains the filter (3) near the top of the frame and some impulses near the bottom boundary. The second frame in Figure 3 is the result of deconvolution by the filter (3). Notice that deconvolution turns the filter itself into an impulse, while it turns the impulses into comet-like images. The use of a helix is evident by the comet images wrapping around the vertical axis.

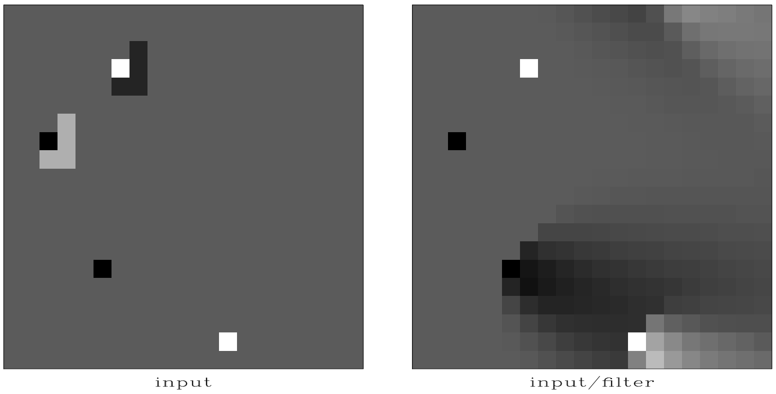

Figure 3: Illustration of 2-D deconvolution. Left is the input. Right is after deconvolution with the filter (3) $\boxed{\text{jon1-wrap}}$ [ER]

In Figure 4, many inputs are tested. Starting from the left are a low-pass blob, a Ricker wavelet, the filter (3) itself, and a couple impulses, one near the bottom boundary. The second frame shows deconvolution by the filter (3). The third frame compounds the second frame with an adjoint (reverse time) deconvolution. (Instead of blowing plumes to the right, it blows them to the left.) The fourth frame convolves the result with the original filter and its adjoint; and we see we are back where we started. No errors, no evidence remains of any of the boundaries where we have wrapped and truncated!

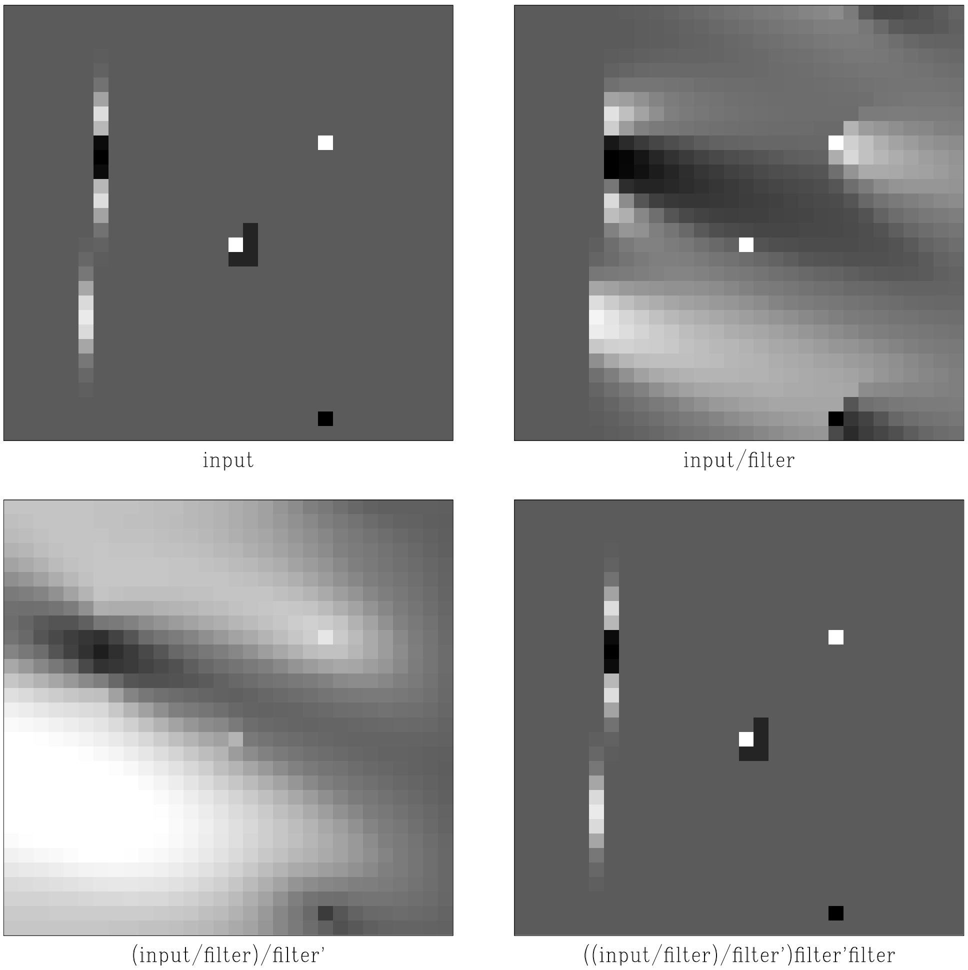

Figure 4: Recursive filtering backwards (leftward on the space axis) is done by the *adjoint* of 2-D deconvolution. Here we see that 2-D deconvolution compounded with its adjoint is exactly inverted by 2-D convolution and its adjoint. jon1-pdadj [ER]

I found myself with a powerful convolution/deconvolution program, but I did not have a bag full of 2-D filters. Further, I knew that a great number of filters are unstable for deconvolution. I recalled an ancient 1-D theorem that if the sum of the absolute values of the filter coefficients after the onset pulse is less than the pulse, then the filter is "positive real," hence "minimum phase," hence stable in polynomial division. Ironically, it is the "almost unstable" filters that hold the greatest interest, because they are the ones that move energy long distances. To make long impulse-responses of deconvolution, I chose all the adjustable filter coefficients to be negative and to sum to -1, as does (3). For damping, I took the absolute value of the sum to be a little less than 1.

In seismology we often have occasion to steer summation along beams. Such an impulse response is shown in Figure 5.

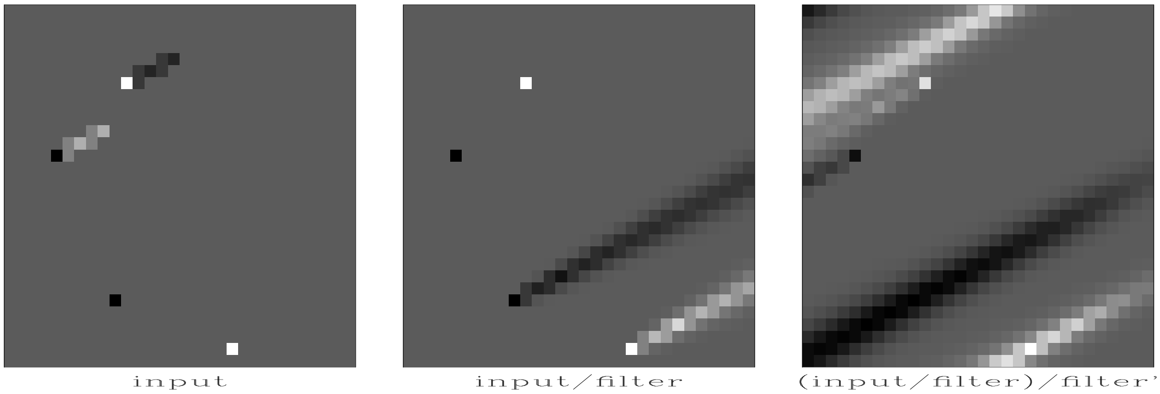

Figure 5: This filter is my guess at a simple low-order 2-D filter whose inverse times its inverse adjoint, is approximately a dipping seismic arrival. jon1-dip [ER]

Finally, I have long had an interest in filters that would destroy plane waves. The inverse of such a filter creates plane waves. A filter that creates two plane waves is illustrated in figure 6.

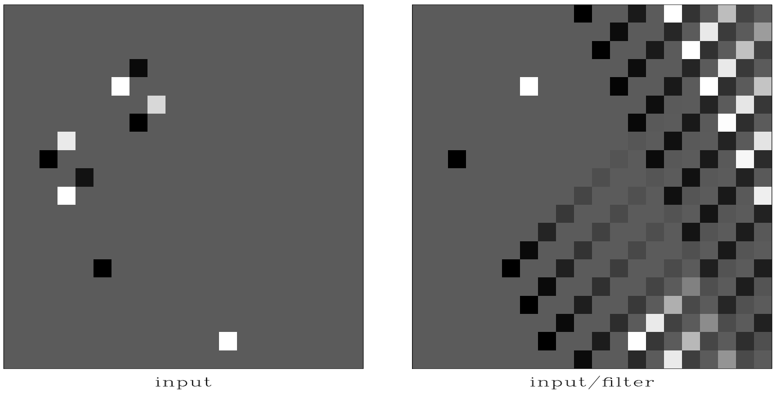

Figure 6: This filter is my guess at a simple low-order 2-D filter whose inverse contains plane waves of two different dips. One of them is spatially aliased. jon1-waves [ER]

PROGRAM FOR MULTIDIMENSIONAL DE/CONVOLUTION

Let us unroll the filter helix seen in Figure 2 and see what we have. Start from the idea that a 2-D filter is generally made from a cluster of values near one another in two dimensions as in the figure.

Notice a "1" at the filter's onset (a common convention) and other values damping to zero a little distance away from the "1". Unrolling this filter into a 1-D strip, we find the "1" on the end of the strip, a few other values nearby, a long sequence of zeros, zeros that were on the back side of the coil, and eventually we come upon some more small values, values that are close to the "1" in the next column of the two-dimensional space, but are far away in one-dimensional space. Thus, in the helical approach, a 2-D filter is a 1-D filter containing some long intervals of zeros. The intervals are the length of a 1-D seismogram.

My program for 2-D convolution with a 1-D convolution program, could convolve with the somewhat long 1-D strip, but it is much more cost effective to ignore the many zeros, which is what I do. I do not multiply by the backside zeros, nor do I even store them in memory. Subroutine `helicon()` did the convolution job for Figure 1. It looks like a one-dimensional convolution program except where the input is shifted from the output. Let a counter `ia` range over the nonzero filter coefficients. Whereas an ordinary convolution program would do time shifting by a code line like `ix=iy-lag`, subroutine `helicon()` ignores the many zero filter values on backside of the tube by using the code `ix=iy-lag(ia)`. Before subroutine `helicon()` is invoked, we need to prepare two lists, one list containing nonzero filter coefficients `aa(ia)`, and the other list containing the corresponding lags `lag(ia)` measured to include multiple wraps around the helix.

You can ignore the second half of subroutine `helicon()` which does adjoint filtering. Because I solve so many least-squares problems, in my textbooks I always accompany every linear operator by its adjoint (matrix transpose). As you might notice, the adjoint of filtering is filtering backwards. The ends of the 1-D axis are a little trickier than you might imagine, but there is no need to examine them now. (One end is transient and the other end is truncated.)

```
# Polynomial multiplication and its adjoint.
#      Designed for gapped filters (helical 2-D, 3-D, etc).
#      Requires the filter be causal with an implicit "1." at the onset.
#
subroutine helicon( adjoint, aa,na, lag,     xx,n123,  yy   )
integer iy, ix, ia, adjoint,    na, lag(na),    n123
real                        aa(na),        xx(n123), yy(n123)
do ia = 1, na
        if( lag(ia) <= 0 ) call erexit('Filter is non causal')
if( adjoint == 0)
        do iy = 1, n123 {
                        yy(iy) = xx(iy)
                do ia = 1, na {                         ix = iy - lag(ia)
                                                    if( ix < 1 ) break
                        yy(iy) = yy(iy) + aa(ia) * xx( ix)
                        }
                }
else
        do ix = n123, 1, -1  {
                        xx(ix) = yy(ix)
                do ia = 1, na {                         iy = ix + lag(ia)
```

```
                                        if( iy > n123 ) break
                        xx(ix) = xx(ix) + aa(ia) * yy( iy)
                        }
                }
return; end
```

FEATURES OF 1-D THAT APPLY TO MANY DIMENSIONS

Time-series analysis is rich with concepts that the helix now allows us to apply to many dimensions. First is the notion of an impulse function. Observe that an impulse function on the surface of the helical cylinder maps to an impulse function on the line of the unwound coil. An autocorrelation function that is an impulse corresponds both to a white spectrum in 2-D and to a white spectrum in 1-D.

The prediction-error-filter (PEF) unites many well established concepts in time-series analysis:

- Subtracting a time series from its prediction yields the prediction error.
- The method of least-squares is used to find the prediction filter. This is also called "autoregression".
- Textbooks such as PVI show that the spectrum of the output of the PEF tends towards whiteness (as the filter length increases). Thus the spectrum of the PEF tends to the inverse of that of the input.
- A time series can be decomposed into random impulses (white spectrum) convolved with a natural wavelet that is the inverse of the PEF.
- For any power spectrum, there is a causal wavelet (with that spectrum) that can be found by "spectral factorization". In the frequency domain this is known as the Kolmogoroff method.
- The PEF has the property of "minimum phase" which means that both it and its convolutional inverse are causal, and this means we have stable recursions.
- Stable filters can be modeled as layered media where waves resonate among reflection coefficients bounded by unity. Such models help in PEF estimation (Burg spectral method).

In summary, the (one-sided) PEF has magical mathematical properties and stable recursions. Symmetrical filters cannot be used recursively and do not have white outputs (which limits their usefulness). Therefore, let us use the helix idea to examine the two-dimensional manifestation of a PEF. For clarity, I adopt the convention that the zero-lag response of the one-dimensional PEF has the value "1". In one dimension, there are zeros before the "1" and adjustable values after it. Figure 2 shows such a filter wrapped on a helix. For most cases of interest, the significant filter coefficients cluster near the "1" and decay with distance (something like most autocorrelation functions). Supposing that nonzero filter coefficients lie within a short distance (two lags) from the "1", we can extract and display the coefficients of the 2-D PEF like this:

$$\begin{array}{ccc} h & c & 0 \\ p & d & 0 \\ q & e & \mathbf{1} \\ s & f & a \\ u & g & b \end{array} \tag{4}$$

The adjustable values in the PEF, $a, b, c, \cdots, u$ are generally found by the method of least squares to minimize the power out of the PEF. Observe that the PEF has the "**1**" along its side, not at a corner (as might be guessed by the analogy of a one-dimensional PEF being one-sided). That a 2-D PEF has its "1" along its side is not a new result. It is in my textbook PVI, but its proof there is abstract and subtle so it is little known. The helix idea makes it far more clear that a 2-D PEF has its "1" along one of its sides.

THE HELIX AND FINITE DIFFERENCES

Discretize the (x, y)-plane to an $N \times N$ array and pack the array into a vector of N^2 components. Likewise pack the Laplacian operator $\partial_{xx} + \partial_{yy}$ into a matrix. For $N = 4$ that matrix is shown in (5).

$$
\left[\begin{array}{cccc|cccc|cccc|cccc}
-4 & 1 & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
1 & -4 & 1 & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & 1 & -4 & 1 & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & 1 & -4 & h & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\hline
1 & \cdot & \cdot & h & -4 & 1 & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & 1 & \cdot & \cdot & 1 & -4 & 1 & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & 1 & \cdot & \cdot & 1 & -4 & 1 & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & 1 & \cdot & \cdot & 1 & -4 & h & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot \\
\hline
\cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & h & -4 & 1 & \cdot & \cdot & 1 & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & 1 & -4 & 1 & \cdot & \cdot & 1 & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & 1 & -4 & 1 & \cdot & \cdot & 1 & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & 1 & -4 & h & \cdot & \cdot & 1 \\
\hline
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & h & -4 & 1 & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & 1 & -4 & 1 & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & 1 & -4 & 1 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & 1 & -4
\end{array}\right] \tag{5}
$$

The two-dimensional matrix of coefficients for the Laplacian operator is shown in (5), where, on a cartesian space, $h = 0$, and in the helix geometry, $h = 1$. (A similar partitioned matrix arises from packing a cylindrical surface into a 4×4 array.) Notice that the partitioning becomes transparent for the helix, $h = 1$.

With the partitioning thus invisible, the matrix simply represents one-dimensional convolution and we have an alternative analytical approach, Fourier Transform. We often need to solve sets of simultaneous equations with a matrix similar to (5). A costly method is to factor the matrix into upper and lower triangular form that can be "backsolved" which in this case amounts to recursive filtering.

The Fourier approach is similar but much faster. The (negative of the) Laplacian operator is regarded as an autocorrelation $(\cdots,-1,\cdots,-1,4,-1,\cdots,-1,\cdots)$. Using the Kolmogoroff spectral-factorization method, a "minimum-phase" wavelet is found which has this autocorrelation. This 2-D wavelet (rotated 90° from (4)) is

$$\begin{array}{cccccccc} & & & & 1.000 & -.363 & -.024 & -.013 & \cdots \\ \cdots & -.024 & -.048 & -.113 & -.312 & & & & \end{array} \tag{6}$$

The 2-D autocorrelation of (6) is the 2-D Laplace operator. The factorization (6) is nonunique in that causality can be defined along either x or y axes; thus the operator (6) can be rotated or reflected. Unfortunately, we see that the factored operator has an infinite number of nonzero terms, but fortunately they seem to be converging rapidly so truncation seems reasonable. We can use this feedback operator to solve Poisson's equation very rapidly. Using the seven coefficients shown, the cost is fourteen multiplications (because we need to run both ways) per mesh point. An example is shown in Figure 7.

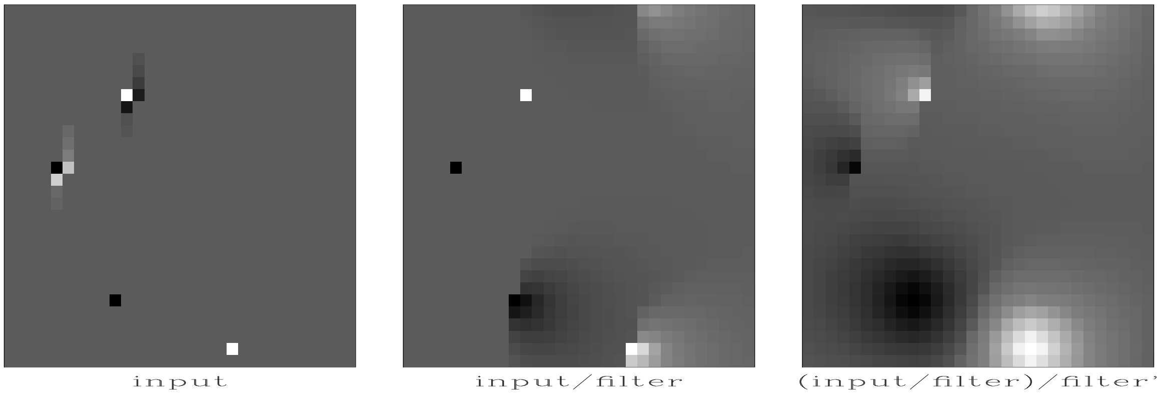

Figure 7: Deconvolution by a filter whose autocorrelation is the two-dimensional Laplacian operator. This amounts to solving the Poisson equation. jon1-lapfac [ER]

As a practical matter, this Poisson equation solver is a convenient isotropic smoothing filter.

Twenty years ago when migrations were two-dimensional, we developed powerful wave-equation methods for finite-difference downward-continuation of wave fields. These methods were frustrated in 3-D because we could not rapidly solve algebraic systems like $[1-\alpha(\delta_{xx}+\delta_{yy})]P(z+\Delta z) = \text{known}$. The helix should resuscitate these methods (although we will need to develop approximations for when $\alpha(x,y)$).

WHY THE RECURSIVE MULTIDIMENSIONAL PEF?

Geophysicists estimate multidimensional functions, especially the earth's impedance and velocity. We are often required to produce such functions to fill maps and 3-D volumes while our measurements provide us with only part of the information we need. Thus we are left with unavoidable problems such as "null space" and/or "empty bins". We should fill this void with "something that looks like everything else." We should produce a gentle transition from the known to the unknown, preserving the statistical properties (spectrum) while we do so. This is also called "hiding the data acquisition footprint." In one dimension we have the frequency spectrum and in higher dimensions we have also the dip spectrum and the velocity spectrum. These ideas are gathered in my book PVI. They have not yet reached industrial-scale practice, perhaps because of the slow rate of convergence of optimization methods such as conjugate gradients (CG) when applied to massive problems.

When I tried optimization methods (such as CG) to achieve "minimum wiggliness" I found I was solving differential equations, not efficiently by using integration (recursion), but inefficiently by applying differential operators millions of times. This suggests that unadorned CG is far from optimum and that huge gains should be possible. This raised my interest in using recursive methods. But how can we use recursion in multidimensional space?

Signal analysis and filter theory provide us with a wealth of knowledge of one-dimensional transformations, but the basic ideas, poles and zeros, appear to fail us in two dimensions, because mathematics teaches us little about polynomials in two variables. In one dimension, it is easy to write a convolution program and its inverse, and we have adequate theory to assure stability of the inverse. Furthermore, we can easily find such operators and their inverses for most spectra of interest. In two dimensions, I found I could not name a single roughening operator for which I knew a reasonably efficient inverse operator. For example, it is easy to speak of the inverse of combinations of one-dimensional derivatives, but hopeless in two dimensions where the gradient operator has no obvious inverse operator and where finding the inverse of the Laplacian operator amounts to solving Laplace's equation, a significant chore.

It is easy to code a two-dimensional convolution program. It amounts manipulating the coefficients of a two-dimensional polynomial. This suggests that we prepare a program for polynomial division by a two-dimensional polynomial. It is easy, easy only so long as we limit ourselves to 2-D filters with the "1" on the corner. A 2-D PEF, however, has its "1" along a side. I was unable to write a polynomial-division for such a filter until I fell upon the idea of helical boundary conditions. A few short hours after I came upon the helical-boundary trick, I understood that a longstanding obstacle, the stability of 2-D deconvolution, would be overcome too. Thus my hopes grew to their present size, hopes for bringing many theoretical estimation techniques up to industrial-scale projects.

Several pilot projects are well underway. The heart of the matter is using multidimensional recursive filters as preconditioners to solve geophysical data fitting problems far more rapidly. Early results are very encouraging. I plan to publish them with my coworkers, Sergey Fomel and Bob Clapp.

REFERENCES

The basic references are my free books, available at my classroom web site: http://sepwww.stanford.edu/sep/prof/

1. For the Burg method, see my first book, FGDP.
2. For Z-transforms, see my third book, PVI.
3. For conjugate-direction, map-estimation examples, see my fifth book, GEM (formerly TDF).