Multidimensional recursive filters via a helix

Jon Claerbout

(presented at the SEG research workshop, Vail, CO, August 3-7, 1997)

Stanford Exploration Project, Mitchell Bldg., Department of Geophysics,

Stanford University, Stanford, CA 94305-2215

(May 21, 1998)

ABSTRACT

Wind a wire onto a cylinder to create a helix. 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. I show some 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 are general-purpose preconditioners that vastly speed the solution of a wide class of geophysical estimation problems. The helix transformation also enables us the partial-differential equation of wave extrapolation as though it was an ordinary-differential equation.

INTRODUCTION

This paper introduces and expounds a basic principle that promises wide-ranging practical applications: A multiple-dimensional Cartesian space can very often be analyzed as a one-dimensional space. Many geophysical image and volumetric estimation problems appear to be multidimensional, but actually they can be handled much more simply. The idea is to use the helical coordinate system. This paper concentrates on explaining the helix idea and it concludes by pointing towards applications in 3-D migration, velocity estimation, and antialiasing.

To see the tip of the iceberg, consider this example: On a two-dimensional Cartesian mesh, the function $\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$

has the autocorrelation $\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$.

Likewise, on a one-dimensional Cartesian mesh,

the function $\mathbf{b} = (1, 1, 0, 0, \cdots, 0, 1, 1)$

has the autocorrelation $\mathbf{r} = (1, 2, 1, 0, \cdots, 0, 2, 4, 2, 0, \cdots, 1, 2, 1)$.

The numbers in the one-dimensional world are identical with the numbers in the two-dimensional world! This correspondence is no accident.

The correspondence between 2-D and 1-D is not merely a mapping between the 2-D and 1-D spaces and autocorrelations in those spaces. It is also a correspondence for filtering. Figure 1 shows some two-dimensional shapes that are convolved together. The left panel shows an impulse-response function, the center shows some impulses, and the right shows the superposition of responses. A surprising, indeed amazing, fact is that Figure 1 was not computed with a two-dimensional convolution program. It was computed with a one-dimensional computer program. It could have been done with anybody's one-dimensional convolution program, either in the time domain or in the Fourier domain. This magical trick is done with the helical coordinate system.

The correspondences above are not limited to two dimensions. They also link multidimensional space to one-dimensional space. Although we can perform multidimensional filtering and spectral operations in one dimension, the gain is only conceptual. The cost is the same.

Breakthroughs arise, however, in concept and in computation when we inspect deconvolution, known to engineers as feedback filtering or recursive filtering, known to numerical analysts as back substitution, and known to physicists as finite differences for partial-differential equations (in Cartesian coordinates). Recursion is a powerful operation but it can be dangerous. With recursion, stability always needs to be considered. Multidimensional recursion would be devoid of power without stability theory and the useful techniques that go with it. Fortunately, the correspondence between one and many dimensions gives us the needed stability theory and techniques.

THE HELIX FILTERING IDEA

Computational work on a 2-D Cartesian mesh sometimes requires a Fourier transform on one of the axes. A byproduct is that the original plane becomes a cylinder—no real problem when the diameter of the cylinder is large enough. The helix idea involves a similar compromise. Take the Cartesian mesh to be a collection of columns. Instead of connecting the bottom of each column to its top getting a cylinder, the helix connects the bottom of one column to the top of the next. The columns all connect into a supercolumn, a one-dimensional space. The mesh can be parameterized either as a 1-D space or as a 2-D space.

The first surprising thing is what the helix does to convolutions and partial differential equations.

A basic idea of filtering, be it in one dimension, two dimensions, or more, is that you have some filter coefficients (finite-differencing star) 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 then summing the terms. Figure 2 shows a helical mesh for 2-D data on a cylinder. Darkened squares depict a 2-D filter shaped like the Laplacian operator. The input data, the filter, and the output data are all on helical meshes all of which could be unrolled into linear strips. A compact 2-D filter like a Laplacian, on a helix is a sparse 1-D filter having long empty gaps.

Since the values output from filtering can be computed in any order, we can slide the filter coil over the data coil in any direction. The order that you produce the outputs is irrelevant. You could compute the results in parallel. 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. We can do 2-D convolution with a 1-D convolution program.

Convolution creates an output q_t from an input p_t and a filter b_τ with

$$q_t \quad = \quad p_t + \sum_{\tau>0} b_\tau p_{t-\tau} \tag{1}$$

(I use the convention that $b_0 = 1.0$). Deconvolution (polynomial division) can undo convolution (polynomial multiplication) by backsolving, by simply rearranging the terms

$$p_t \quad = \quad q_t - \sum_{\tau>0} b_\tau p_{t-\tau} \tag{2}$$

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 (because an ordinary filter makes outputs from past *inputs* whereas a recursive filter makes them from past *outputs*).

Recursive filtering sometimes solves big problems with astonishing speed. It can propagate energy rapidly for long distances. Unfortunately, recursive filtering can also be unstable. The most interesting case, near resonance, is also near instability. There is an old textbook literature e.g. Claerbout (1976) or (1992) with extensive technology for recursive filtering in one dimension. The helix allows us to apply that technology to two (and more) dimensions. It is a wonderful insight. We could not previously do 2-D deconvolution because we had no stability theory for it. We cannot simply use polynomial division to undo the 2-D Laplacian operator for example, because the output will diverge. Poles and zeros tell us about 1-D stability but we don't have them for 2-D polynomials.

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. Vector-valued time signals meaning multichannel time series (Claerbout, 1976) appear to be related to a scalar function of the space axes, but that relationship is not helpful here.

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 two copies of 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.

Using the result of Figure 3 as an input, the first frame in Figure 4 results from backwards recursion, filtering backwards along the helix, the adjoint operator $\mathbf{F}'$. The adjoint blows energy leftward. Filtering forward and backwards has implemented a symmetrical smoothing operator $(1/\mathbf{F})/\mathbf{F}'$. The last frame in Figure 4 shows that the two deconvolution smoothers can be inverted by convolution, convolving once with filter (3) and once with its adjoint (reverse on both axes). We see we are back where we started. No errors, no evidence remains of any of the boundaries where we have wrapped and truncated. Using the helix, we find that one-dimensional polynomial division is a rapidly invertible *multidimensional* transformation, one with many adjustable parameters.

PROGRAM FOR MULTIDIMENSIONAL 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 similar to the Laplacian operator in the figure. We see that 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.

A 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 we do. We do not multiply by the backside zeros, nor do we even store them in memory. Whereas an ordinary convolution program would do time shifting by a code line like `ip=iq-lag`, subroutine `helicon()` ignores the many zero filter values on backside of the tube by using the code `ip=iq-lag(ib)` where a counter `ib` ranges over the nonzero filter coefficients. Before subroutine `helicon()` is invoked, we need to prepare two lists, one list containing nonzero filter coefficients `bb(ib)`, and the other list containing the corresponding lags `lag(ib)` measured to include multiple wraps around the helix. For example, the 2-D Laplace operator

$$\begin{array}{ccc} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{array} \tag{4}$$

can be thought of as the 1-D filter

$$1, 0, \cdots, 0, 1, -4, 1, 0, \cdots, 0, 1, 0, \cdots \tag{5}$$

The first filter coefficient (at lag 0) in the filter (5) is +1.0 as is implicit to subroutine `helicon()`. To apply the remaining Laplacian coefficients on a 1000×1000 mesh requires the two four-component vectors:

```
i    lag(i)   bb(i)
---  ------   -----
1      999      1.0
2     1000     -4.0
3     1001      1.0
4     2000      1.0
```

Subroutine `helicon()` did the convolution job for Figure 1. The second half of subroutine `helicon()` does adjoint filtering. The adjoint of filtering is filtering backwards. [The ends of the 1-D axis are a little tricky (one end transient, the other truncated) but the code passes the dot-product test for adjoints $\mathbf{q}'(\mathbf{B}\mathbf{p}) = (\mathbf{B}'\mathbf{q})'\mathbf{p}$.]

```
# 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, bb,nb, lag,     pp,n123,  qq   )
integer iq, ip, ib, adjoint,    nb, lag(nb),    n123
real                         bb(nb),         pp(n123), qq(n123)
do ib = 1, nb
        if( lag(ib) <= 0 ) call erexit('Filter is non causal')
if( adjoint == 0)
        do iq = 1, n123 {
                        qq(iq) = pp(iq)
                do ib = 1, nb {                       ip = iq - lag(ib)
                                                  if( ip < 1 ) break
                        qq(iq) = qq(iq) + bb(ib) * pp( ip)
                        }
```

```
        }
else
    do ip = n123, 1, -1  {
            pp(ip) = qq(ip)
        do ib = 1, nb {                    iq = ip + lag(ib)
                                         if( iq > n123 ) break
            pp(ip) = pp(ip) + bb(ib) * qq( iq)
            }
        }
return; end
```

The companion program for deconvolution is virtually identical, except for the obvious differences between equations (1) and (2).

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 2-D surface of the helical cylinder maps to an impulse function on the 1-D line of the unwound coil. An autocorrelation function that is an impulse corresponds both to a white (constant) spectrum in 1-D and to a white (constant) spectrum in 2-D. A causal filter in one dimension has a curious shape on the two-dimensional helix. In one dimension, the causal filter has zeros before the "1.0" and various values after it. Say the nonzero filter coefficients on the cylinder lie within a short distance (two lags) from the "1.0". Extract the little 2-D patch (which is the end of the 1-D filter). I display it reversed on both axes so the reader can envision it as crosscorrelation, first moving down the first seismogram and then down the next.

$$
\begin{array}{ccc} u & g & b \\ s & f & a \\ q & e & \mathbf{1} \\ p & d & 0 \\ h & c & 0 \end{array}
\quad = \quad
\begin{array}{ccc} u & g & b \\ s & f & a \\ q & e & \cdot \\ p & d & \cdot \\ h & c & \cdot \end{array}
\quad + \quad
\begin{array}{ccc} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \mathbf{1} \\ \cdot & \cdot & 0 \\ \cdot & \cdot & 0 \end{array}
\qquad (6)
$$

where $a, b, c, ..., u$ are adjustable coefficients. Thus we conclude that the 2-D analog of a 1-D causal filter has its abrupt beginning along the side of the 2-D filter.

A special causal filter that unites many well established concepts in time-series analysis is the prediction-error-filter (PEF). A 2-D PEF, like a 1-D PEF, is a causal filter with adjustable coefficients as in the array (6), that are adjusted to minimize the filter's output energy (for a particular input signal). That the 2-D PEF should have its beginning along a side (instead of at a corner) is an abstract idea that I have always found difficult to teach clearly, until I fell upon the helix explanation.

Here is a brief summary of important ideas in time-series analysis that the helix makes applicable in higher dimensions:

- The filter $(a, b, c, ..., u)$ is the negative of the prediction filter. The filter $(1, a, b, c, ..., u)$ is the prediction-error filter.
- The method of least-squares is used to find the prediction filter. This is also called "autoregression".
- Textbooks (Claerbout, 1998) show that the spectrum of the output of the PEF tends towards whiteness as the filter length increases. Thus the spectrum of the PEF itself

tends to the inverse of that of its input. (Noncausal filters do not have white outputs and cannot be used recursively.)

- 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 (1939) method.
- The PEF has the property of "minimum phase" which means that both it and its convolutional inverse are causal. Thus, we can design stable multidimensional recursive filters as we do in one dimension.
- Stable filters can be modeled as layered media where waves resonate among reflection coefficients bounded in absolute value by unity. Such models help in PEF estimation by the Burg (1975) method.

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. The helical coordinate system makes a giant step towards overcoming this difficulty. The many features of 1-D theory outlined above are now awaiting multidimensional application.

FINITE DIFFERENCES ON A HELIX

The function

$$\mathbf{r} = (-1, 0, \cdots, 0, -1, 4, -1, 0, \cdots, 0, -1) \tag{7}$$

is an autocorrelation function. It is symmetrical about the "4" and its Fourier transform is positive for all frequencies. Digging out an old textbook (Claerbout, 1976), we discover how to compute a causal wavelet with this autocorrelation. I used the "Kolmogoroff spectral-factorization method" to find this wavelet $\mathbf{h}$:

$$\mathbf{h} \quad = \quad (1.791, -.651, -.044, -.024, \cdots, \cdots, -.044, -.087, -.200, -.558) \tag{8}$$

Wind the signal $\mathbf{r}$ around a vertical-axis helix to see its two-dimensional shape $\mathbf{R}$:

$$\mathbf{R} \quad = \quad \begin{bmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{bmatrix} \tag{9}$$

This 2-D filter is is the negative of the finite-difference representation of the negative of the Laplacian operator, generally denoted $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$. Now wind the signal $\mathbf{h}$ around the same helix to see its two-dimensional shape $\mathbf{H}$:

$$\mathbf{H} \quad = \quad \begin{bmatrix} & & & & 1.791 & -.651 & -.044 & -.024 & \cdots \\ \cdots & -.044 & -.087 & -.200 & -.558 & & & & \end{bmatrix} \tag{10}$$

In the 2-D representation (10) we see the coefficients diminishing rapidly away from maximum value 1.791. My claim is that the 2-D autocorrelation of (10) is (9). You verified this idea at the beginning of this paper where the numbers were all ones. You can check it again in a few moments if you drop the small values, say 0.2 and smaller.

Since the autocorrelation of $\mathbf{H}$ is $\mathbf{H}'\mathbf{H} = \mathbf{R} = -\nabla^2$ is a *second* derivative, the operator $\mathbf{H}$ must be something like a *first* derivative. As a geophysicist, I found it natural to compare the operator $\frac{\partial}{\partial y}$ to $\mathbf{H}$ by applying them to a local topographic map. The result shown in Figure 5 is that $\mathbf{H}$ enhances drainage patterns whereas $\frac{\partial}{\partial y}$ enhances mountain ridges.

Our construction makes $\mathbf{H}$ have the energy spectrum $k_x^2 + k_y^2$, so the magnitude of its

Fourier transform is $\sqrt{|k_x^2 + k_y^2|}$. This rotationally invariant function in the Fourier domain contrasts sharply with the nonrotationally invariant function shape in (x, y)-space. The difference must arise from the phase spectrum. The factorization (10) is nonunique in that causality associated with the helix mapping can be defined along either x or y axes; thus the operator (10) can be rotated or reflected.

The operator $\mathbf{H}$ has curious similarities and differences with the familiar gradient and divergence operators. In two-dimensional physical space, the gradient maps one field to *two* fields (north slope and east slope). The factorization of $-\nabla^2$ with the helix gives us the operator $\mathbf{H}$ that maps one field to *one* field. Being a one-to-one transformation (unlike gradient and divergence) the operator $\mathbf{H}$ is potentially invertible by deconvolution (recursive filtering).

I have chosen the name[1] "helix derivative" or "helical derivative" for the operator $\mathbf{H}$. A telephone pole has a narrow shadow behind it. The helix integral (middle frame of Figure 6) and the helix derivative (left frame) show shadows with an angular bandwidth approaching 180°. Thus, $\mathbf{H}$ is much less directional than either $\frac{\partial}{\partial x}$ or $\frac{\partial}{\partial y}$.

This is where the story all comes together. One-dimensional theory, such as the Kolmogoroff spectral factorization, produces not merely a causal wavelet with the required autocorrelation. It produces a wavelet that is stable in deconvolution. Using $\mathbf{H}$ in one-dimensional polynomial division, we can solve many formerly difficult problems very rapidly. Consider the Laplace equation with sources (Poisson's equation). Polynomial division and its reverse (adjoint) gives us $\mathbf{p} = (\mathbf{q}/\mathbf{H})/\mathbf{H}'$ which means that we have solved $\nabla^2 \mathbf{p} = -\mathbf{q}$

[1] Any fact this basic should be named in some earlier field of mathematics or theoretical physics. Admittedly, the concept exists on an infinite Cartesian plane without a helix, but all my codes in a finite space involve the helix, and the helix concept led me to it.

in Figure 6 by using polynomial division on a helix. Using the seven coefficients shown, the cost is fourteen multiplications (because we need to run both ways) per mesh point.

Matrix view of the helix

Finally, let us look at the helix from the view of matrices and numerical analysis. This is not easy because the matrices are so large. Discretize the (x, y)-plane to an $N \times M$ array and pack the array into a vector of $N \times M$ components. Likewise pack the Laplacian operator

$\partial_{xx} + \partial_{yy}$ into a matrix. For a 4×3 plane, that matrix is shown in equation (11).

$$
-\nabla^2 \quad = \quad \left|\begin{array}{cccc|cccc|cccc}
4 & 1 & \cdot & \cdot & -1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
-1 & 4 & -1 & \cdot & \cdot & -1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & -1 & 4 & -1 & \cdot & \cdot & -1 & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & -1 & 4 & h & \cdot & \cdot & -1 & \cdot & \cdot & \cdot & \cdot \\
\hline
-1 & \cdot & \cdot & h & 4 & -1 & \cdot & \cdot & -1 & \cdot & \cdot & \cdot \\
\cdot & -1 & \cdot & \cdot & -1 & 4 & -1 & \cdot & \cdot & -1 & \cdot & \cdot \\
\cdot & \cdot & -1 & \cdot & \cdot & -1 & 4 & -1 & \cdot & \cdot & -1 & \cdot \\
\cdot & \cdot & \cdot & -1 & \cdot & \cdot & -1 & 4 & h & \cdot & \cdot & -1 \\
\hline
\cdot & \cdot & \cdot & \cdot & -1 & \cdot & \cdot & h & 4 & -1 & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & -1 & \cdot & \cdot & -1 & 4 & -1 & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & \cdot & \cdot & -1 & 4 & -1 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & \cdot & \cdot & -1 & 4
\end{array}\right| \tag{11}
$$

The two-dimensional matrix of coefficients for the Laplacian operator is shown in (11), 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 4×3 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, one-dimensional Fourier Transform. We often need to solve sets of simultaneous equations with a matrix similar to (11). The method we use is triangular factorization.

Although the autocorrelation $\mathbf{r}$ has mostly zero values, the factored autocorrelation $\mathbf{h}$ from (8) has a great number of nonzero terms, but fortunately they seem to be converging rapidly (in the middle) so truncation (of the middle coefficients) seems reasonable. I wish I could show you a larger matrix, but all I can do is to pack the signal $\mathbf{h}$ into shifted columns of a lower triangular matrix $\mathbf{H}$ like this:

$$
\mathbf{H} \;=\; \left[\begin{array}{cccccccccccc}
1.8 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
-.6 & 1.8 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
0.0 & -.6 & 1.8 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
-.2 & 0.0 & -.6 & 1.8 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
-.6 & -.2 & 0.0 & -.6 & 1.8 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & -.6 & -.2 & 0.0 & -.6 & 1.8 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & -.6 & -.2 & 0.0 & -.6 & 1.8 & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & -.6 & -.2 & 0.0 & -.6 & 1.8 & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & -.6 & -.2 & 0.0 & -.6 & 1.8 & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & -.6 & -.2 & 0.0 & -.6 & 1.8 & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -.6 & -.2 & 0.0 & -.6 & 1.8 & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -.6 & -.2 & 0.0 & -.6 & 1.8
\end{array} \right] \qquad (12)
$$

If you will allow me some truncation approximations, I now claim that the laplacian represented by the matrix in equation (11) is factored into two parts $-\nabla^2 = \mathbf{H}'\mathbf{H}$ which are upper and lower triangular matrices whose product forms the autocorrelation seen in (11). Recall that triangular matrices allow quick solutions of simultaneous equations by backsubstitution. That is what we do with our deconvolution program.

GEOESTIMATION: EMPTY BIN EXAMPLE

The basic formulation of a geophysical estimation problem consists of setting up two goals, one for data fitting, and the other for model smoothing. We have data $\mathbf{d}$, a map or model $\mathbf{m}$, a transformation and a roughening filter $\mathbf{A}$, like the gradient ∇ or the helix derivative

$\mathbf{H}$. The two goals may be written as:

$$\mathbf{0} \approx \mathbf{r} = \mathbf{Lm} - \mathbf{d} \tag{13}$$

$$\mathbf{0} \approx \mathbf{x} = \mathbf{Am} \tag{14}$$

which defines two residuals, a data residual $\mathbf{r}$, and a model residual $\mathbf{x}$, which are usually minimized by iterative conjugate-gradient, least-squares methods. A common case is "minimum wiggliness" when $\mathbf{A}$ is taken to be the gradient operator. The gradient is not invertable so we cannot precondition with its inverse. On the other hand, since $-\nabla^2 = \mathbf{div} \cdot \mathbf{grad} = \mathbf{H}'\mathbf{H}$, taking $\mathbf{A}$ to be the helix derivative $\mathbf{H}$, we can invert $\mathbf{A}$ and proceed with the preconditioning: We change the free variable in the fitting goals from $\mathbf{m}$ to $\mathbf{x}$ (by inverting (14)) with $\mathbf{m} = \mathbf{A}^{-1}\mathbf{x}$ and substituting into both goals getting new goals

$$\mathbf{0} \approx \mathbf{LA}^{-1}\mathbf{x} - \mathbf{d} \tag{15}$$

$$\mathbf{0} \approx \mathbf{x} \tag{16}$$

In my experience, iterative solvers find convergence much more quickly when the free variable is the roughened map $\mathbf{x}$ rather than the map $\mathbf{m}$ itself.

Figure 7 (left) shows ocean depth measured by a Seabeam apparatus. Locations not surveyed are evident as the homogeneous gray area. Using a process akin to "blind deconvolution" a 2-D prediction error filter $\mathbf{A}$ is found. Then missing data values are estimated and shown on the right. Preconditioning with the helix speeded this estimation by a factor of about 30. The figure required a few seconds of calculation for about 10^5 unknowns.

Here is how I came to the helix: For estimation theory to be practical in reflection seismology, I felt I needed a faster solution to simple problems like the Seabeam empty-bin problem. To test the idea of a preconditioner using two-dimensional polynomial division, I first coded it with the "1.0" in the corner. Then when I tried to put the "1.0" on the side where it belongs (see (6)), I could not adequately initialize the filter at the boundaries until I came upon the idea of the helix. I had never set out to solve the problem of stability of multidimensional feedback filtering (which I believed to be impossibly difficult). Serendipity.

This example suggests that the philosophy of image creation by optimization has a dual orthonormality: First, Gauss (and common sense) tells us that the data residuals should be roughly equal in size. Likewise in Fourier space they should be roughly equal in size, which means they should be roughly white, i.e. orthonormal. (I use the word "orthonormal" because white means the autocorrelation is an impulse, which means the signal is statistically orthogonal to shifted versions of itself.) Second, to speed convergence of iterative methods, we need a whiteness, another orthonormality, in the solution. The map image, the physical function that we seek, might not be itself white, so we should solve first for another variable, the whitened map image, and as a final step, transform it to the "natural colored" map.

Often geophysicists create a preconditioning matrix $\mathbf{B}$ by inventing columns that "look like" the solutions that they seek. Then the space $\mathbf{x}$ has many fewer components than the space of $\mathbf{m}$. This approach is touted as a way of introducing geological and geophysical prior information into the solution. Indeed, it strongly imposes the form of the solution. This approach is more like "curve fitting" than "inversion." My preferred approach is not to invent the columns of the preconditioning matrix, but to estimate the prediction-error filter of the model and use its inverse.

FURTHER APPLICATIONS

I expect applications to rapidly outpace what I write here (that's been happening during the review process) but anyway I indicate a few new directions.

Velocity estimation resembles the empty-bin problem because data quality provides us with velocity information in some locations but not in others. Where we have no velocity information, we often have an idea of the orientation of bedding planes or we have other dip information that tells us how to interpolate or smooth velocity between measurement locations. Figure 8 shows how easy it is to use the helix to create directional smoothers. Clapp et.al (1998) demonstrate constructing velocity functions via the helix by the use of directional smoothers or "flags." The space-variable flag filter is introduced as the regularization for a conventional velocity estimation process.

Poststack 3-D wave-equation migration is another area ripe for the helix. When 3-D became important, the finite-difference method declined because we were unable to find rapid solutions to large scale algebraic equations with coefficient matrices like (11). A first question to be asked, however, is whether the helix does anything new that we could not do with Fourier transforms? Certainly, much of the above depends on Fourier transforms, and some could have been done with Fourier transforms. Recursive filtering is often faster than Fourier transforms, but the real contribution of the helix depends on a weakness of Fourier transformation. The frequency domain is great for differential equations with constant coefficients, stationary statistics, and regularly sampled data, but its utility diminishes when these conditions are not met. This theme can be developed further into 3-D poststack migration process that handles 3-D velocity variation in a wave-equation consistent manner (Rickett et al., 1998).

Spatial filters easily handle aliased data. Figure 9 shows a filter that destroys two waves, one at the border of aliasing, the other well beyond aliasing. The figure shows that the inverse of such a filter creates aliased plane waves. Claerbout (1992) illustrates synthetic data processing beyond aliasing. Recently Crawley (1998) began demonstrating stacking on field data that is aliased in offset at constant midpoint. Again, the helix should accelerate convergence.

ACKNOWLEDGMENTS

I am delighted to acknowledge many inspirational conversations with Sergey Fomel, who also made Figures 2 and 7. I would like to thank Dave Nichols, Bill Harlan, and Bill Symes for teaching me about preconditioners.

REFERENCES

Burg, J. P., 1975, Maximum entropy spectral analysis: Ph.D. thesis, Stanford University, http://sepwww.stanford.edu/oldreports/sep06/.

Claerbout, J. Fundamentals of Geophysical Data Processing:. http://sepwww.stanford.edu/sep/prof/, 1976.

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

Claerbout, J. Geophysical Estimation by Example: Environmental soundings image enhancement:. http://sepwww.stanford.edu/sep/prof/, 1998.

Clapp, R., Biondi, B., Fomel, S., and Claerbout, J., 1998, Regularizing velocity estimation using geologic dip information: 68th Annual Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts.

Crawley, S., 1998, Shot interpolation for radon multiple suppression: 68th Annual Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts.

Kolmogoroff, A., 1939, Sur l'interpolation et l'extrapolation des suites stationnaires: C.R.Acad.Sci.(Paris), **208**, 2043–2045.

Rickett, J., Claerbout, J., and Fomel, S., 1998, Implicit 3-D depth migration by wavefield extrapolation with helical boundary conditions: 68th Annual Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts.

LIST OF FIGURES

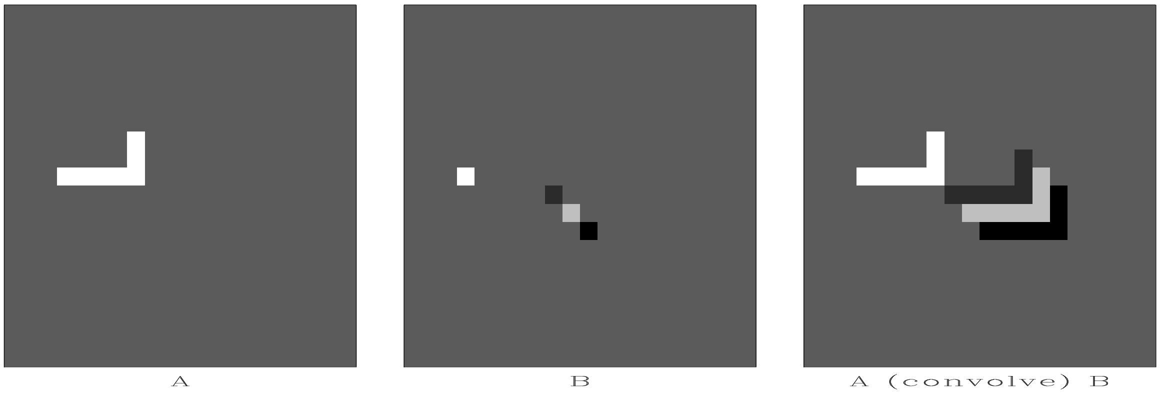

Figure 1.

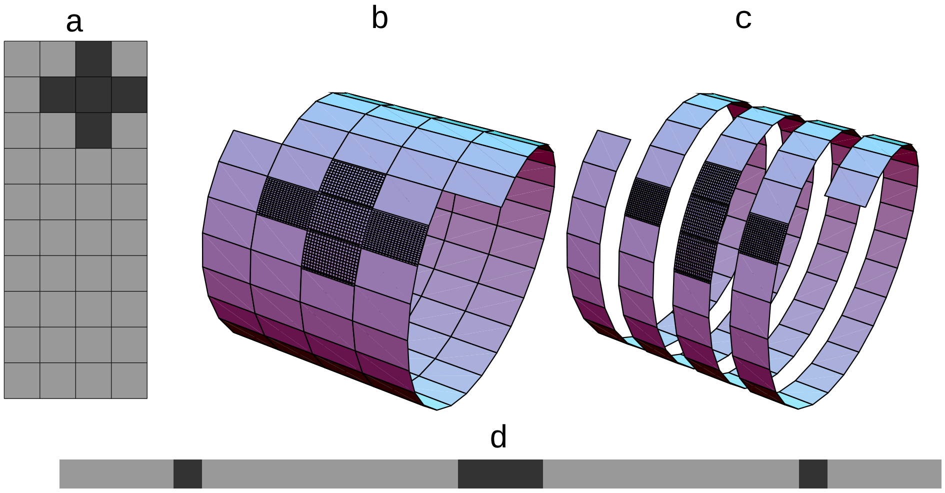

Figure 2.

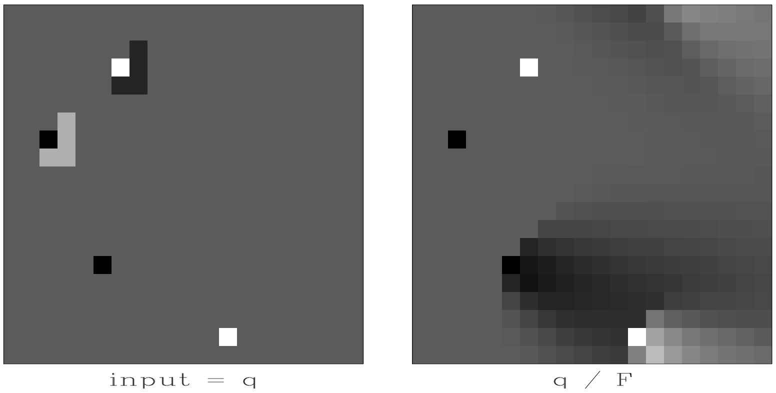

Figure 3.

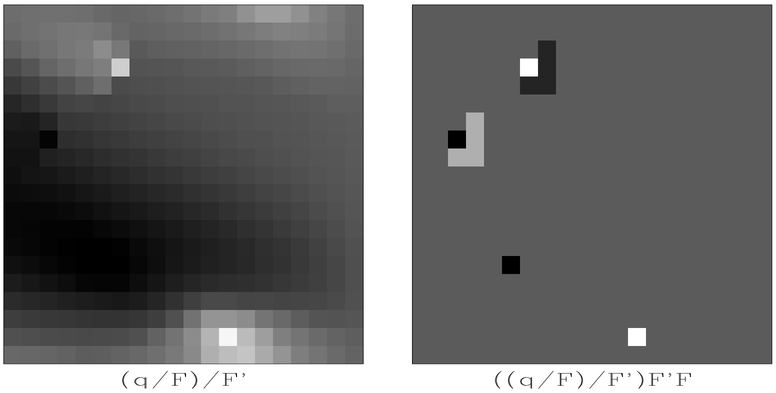

Figure 4.

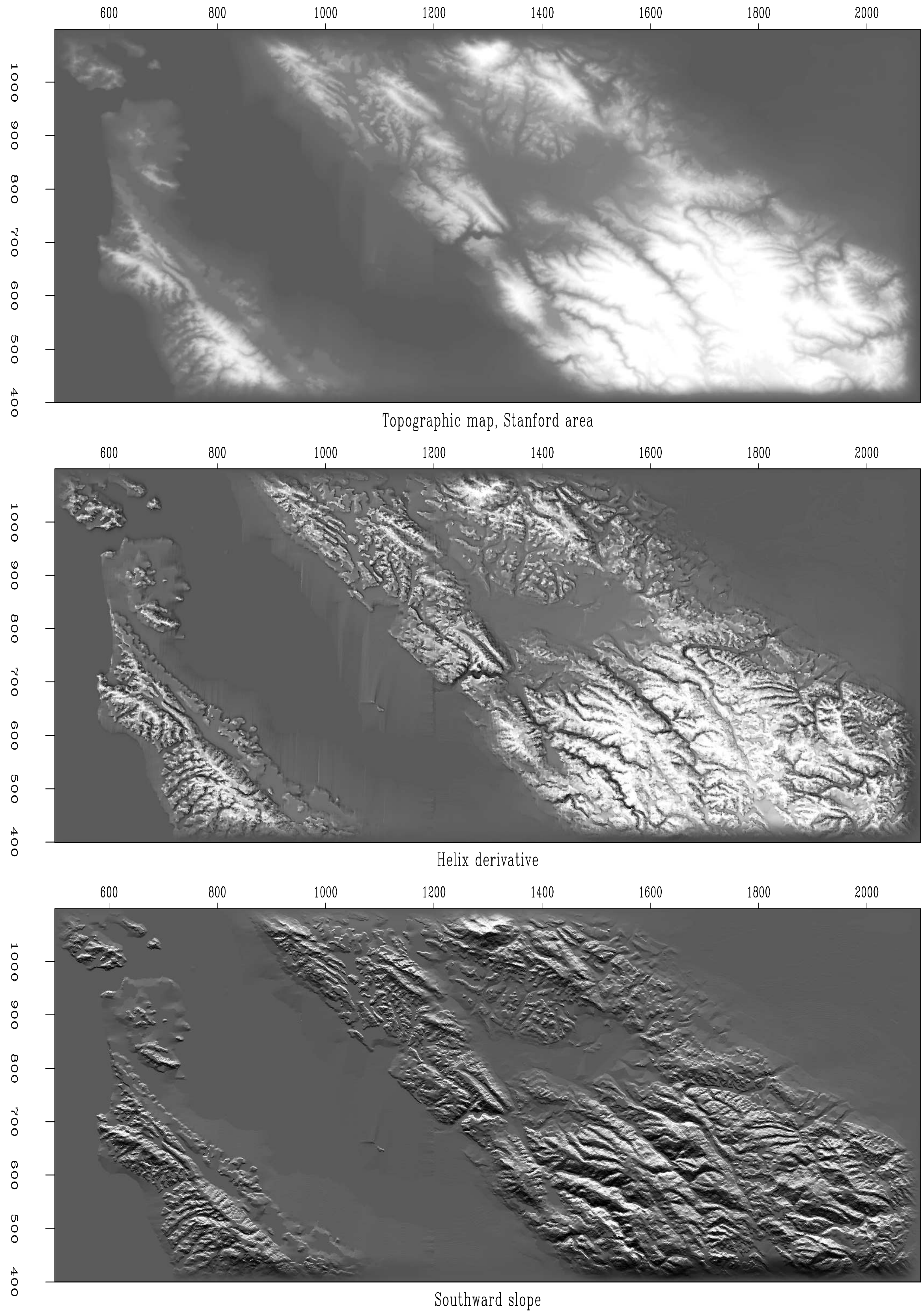
600
800
1000
1200
1400
1600
1800
2000
1000
900
800
700
600
500
400
Topographic map, Stanford area
Helix derivative
Southward slope

Figure 5.

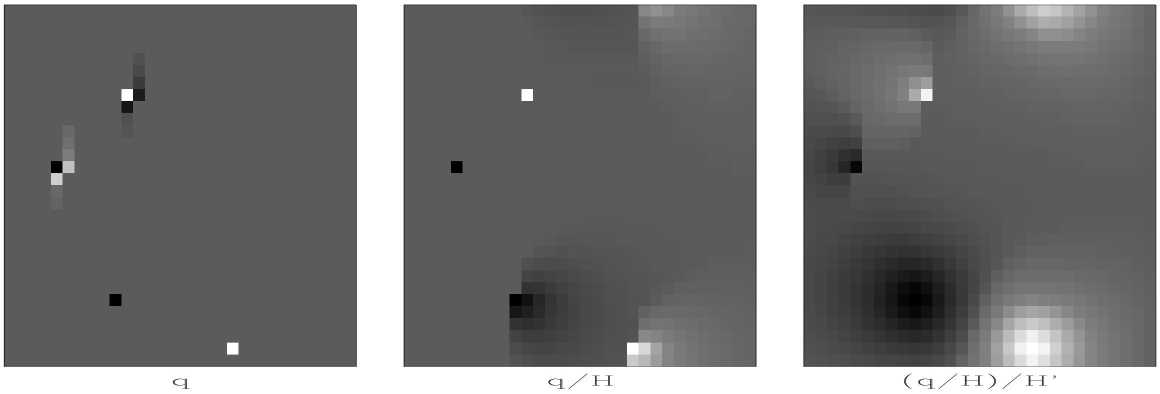

Figure 6.

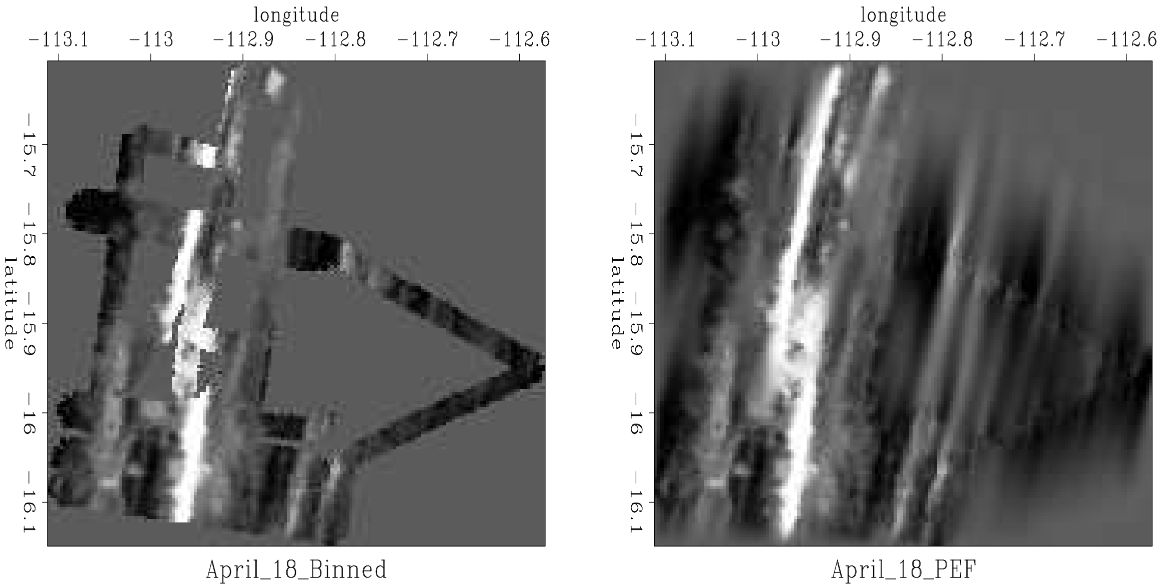
longitude
−113.1
−113
−112.9
−112.8
−112.7
−112.6
latitude
−15.7
−15.8
−15.9
−16
−16.1
April_18_Binned
April_18_PEF

Figure 7.

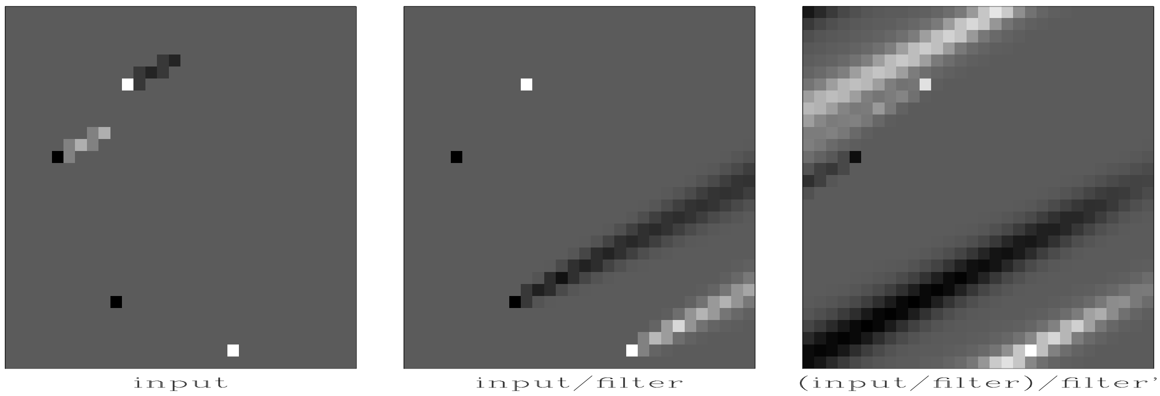

Figure 8.

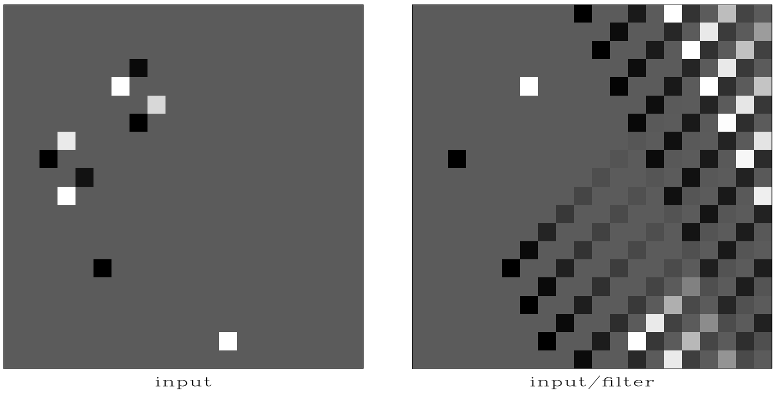

Figure 9.