Interaction with 1-D Seismology

Jon F. Claerbout

ABSTRACT

I wrote a program for user interaction with the basic one-dimensional transforms of seismology and mathematical physics. A signal is displayed in each of two screen windows. The signals are related by a choice of relationships such as Fourier transform or impedance to reflection coefficient. Hand editing on one signal shows up immediately in its effect on the other. The program, called *ed1D*, runs on a Sun workstation.

INTRODUCTION

In SEP 51 I described a program for overlaying hyperbolas on seismic data and performing moveout interactively. The program allows interactive development of a layered model. A small annoyance is that the layers do not always connect smoothly, that is to say, between two bona fide layers you might find a thin layer whose thickness reveals the inaccuracy of your pointing. The program lacks a *signal edit* capability for the velocity/depth model.

In preparing materials for classroom use, I have learned that it is hard to get enough examples. Particularly in Fourier analysis and also in layered media theory, people like to see many examples. With these experiences in mind, I set out to make an interactive one-dimensional signal generator and editor.

As soon as the signal editor was completed, I set up two windows, and a signal in each that can be edited. As a signal in one window is edited, its Fourier transform is updated in the other window. Likewise you can edit the Fourier transform and see the effect on the original signal.

With these capabilities in place, it seemed natural to add most of the other functions that we deal with in layered media seismology, including the theoretical relation between an acoustic well log and a reflection seismogram. Now people can experience themselves the sensitivity of a model to small noises in the data.

Eventually I had 13 signal types. The program serves to investigate the relationship between any pair. Although it could be used for serious work in fitting reflection data to well logs, I designed it more with the student in mind.

Tutorial

On the control panel is a tutorial button. The tutorial takes you through a variety of examples and exercises. I am pleased to report that unlike my earlier hyperbola overlay program, this program has proven to be reliably self teaching. You can throw away this paper document.

HOW YOU EDIT A SIGNAL

There are several means of editing a signal. First you can use the mouse simply to draw the signal you wish. Second, you can draw a weighting function to multiply a signal that you have previously prepared. Third, there are a variety of preexisting analytic signals that you can use as weights. The mouse is used to select adjustable parameters such as the bandwidth on these analytic signals.

Finally, there is a parabolic shaped editing tool that you can *push* against any signal to deform it. Also, you can adjust the curvature of the parabola.

CONTROL PANEL

Globally available controls

The mouse button on the Sun workstation has three buttons. The rightmost button is not used by this program. The leftmost button, under the index finger of a righthanded person, is used to select activities, the middle button generally terminates activities or reverses their sense. As you move the mouse over the control panel buttons, commentary emerges in the message window telling you what will happen if you make the selection. Some of the buttons and their feedback are exhibited below:

TUTOR	Click here with the left mouse button.
dt	horizontal spacing increment: LEFT=shrink. MIDD=expand.
half	ordinate symmetry display: LEFT=all. MIDD=signif half.
feedback	Feedback: LEFT=commentary. MIDD=coordinates.
buffer	LEFT: signal to buffer. MIDD: buffer to signal.
undo	Undo last change in selected view.
hard E	Hardcopy signals end to end.
hard U	Hardcopy signals over and under.

Built-in functions

Push buttons exist for simple functions such as squaring a signal, adding noise, or for making a signal that is all noise. Also, to illustrate sampling phenomena, there is a comb function. The most commonly used push-button built-in signals are the zero signal and the constant signal.

Any signal, such as a sinusoid or a constant signal can be weighted by a Gaussian, exponential, triangle, box, or Cauchy function, and the center and the decay rate can be selected by moving the mouse. Likewise you can truncate a step function with these analytic signals. Since you can initialize a constant signal, the analytic weighting functions can also be made into analytic signals.

SIGNAL SPACES

You select two signal spaces, one for the top window and one for the bottom window. The signal spaces come from the following:

- Distance domain, even
- Wavenumber domain, even
- Causal function of time
- Real(ω) (or Hilbert amplitude)
- Imaginary(ω) (or Hilbert phase)
- Minimum phase wavelet (escaping or earthquake wave)
- logarithm of amplitude spectrum
- Power spectrum
- $r(t)$ = Reflection seismogram (one side of autocorrelation)
- $t\ r(t)$ = time $\times$ 1-D reflection seismogram
- $c(t)$ = reflection coefficients
- $t\ c(t)$ = t-gained "multiple-free" seismogram
- acoustic impedance = density $\times$ material velocity

It was my own choice to display the spaces of $t\,r(t)$ and $t\,c(t)$. As reported in SEP-40 and in my textbook, *Imaging the Earth's Interior,* after studying 40 field profiles from around the world I concluded that t^2 is generally an appropriate gain function for raw data. Of the two powers of t, one is attributed to spherical divergence. The other power of t is some combination of factors such as absorption, scattering, and the effect of gravitational self-compression and the geothermal gradient that diminishes reflectivity with depth. Thus for one-dimensional seismology the appropriate *a priori* display gain is a single power of t.

Topology

Originally I had three disjoint study areas, namely, a Fourier area, a Kolmogoroff area, and a layered media area. Then I realized that these areas overlap, and they are potentially connected. So I connected the study areas in what seems to me to be the most natural way. With the above 13 signals there are $(13 \times 12)/2 = 78$ possible pairs that you could investigate. This is far from all possible relationships in one-dimensional seismology but it does include the relationships of greatest interest. The displays are all real valued. For those spaces where the imaginary part is of interest, I have defined it as a separate signal.

Each signal space is a node in a directed graph. The screen displays any two spaces. After you manually edit one space, the other space is updated through a series of operations shown by the node connections in the graph. The topology is below:

```
Nodes are denoted by ":".
Directed paths have arrowheads like <, >, ^, or v.
Paths without arrowheads go both ways.

    Hilbert amplitude     AA:3   PP:4   Hilbert phase
                            |    /
                            |   /
                            |ft/
                            | /
                            |/
          time, causal    BB:2 ------------------------------|
                            ^                                |
                            |                                |
                            |                                |
                      FT(step(IFT()))                        |
                            |                           IFT(exp(FT()))
                            |                                |
space, even                 |                                |
   XX:0 ----- KK:1 ---=-- UU:6  log spectrum                 |
            wavenumber      ^                                |
                            |                                |
                           log                               |
                            |                                v
            spectrum      SS:7 <------------------------- TT:5
                            |                           minimum phase
                            |                            (escaping wave)
                        ft or ift
                            |         ABOVE calculations by Fourier methods.
                            |         BELOW calculations by recursion.
                            |
 (half) autocorrelation   RR:8 -----------  TR:9    t * r(t)
   reflection seismogram    |                        (reflection data display)
                            |
                            |
reflection coefficients   CC:10 ----------  TC:11   t * c(t)
                            |                        (multiple-free seismogram)
                            |
     acoustic impedance   PV:12
```

I built a routine for each node-to-node transition. Some node pairs can be traversed in both directions. Some paths naturally go only one way, for example, when you multiply by a step function there is no way to go back. Other paths were limited to be one way to ensure that the path from any node to any other is unique. If the path were not unique, then the user would need to specify it. I decided the user would rather not specify the path, and would prefer to specify just the beginning and ending nodes. A message window displays the path as the computations proceed.

Suppose you were comparing a spectrum to a log spectrum. The way I have set it up, the transition to the logarithm is direct, but the inverse transition goes by way of a minimum-phase wavelet. So if you have a spectrum with holes in it, you see time domain truncation errors on the way from the log spectrum back to the spectrum.

(Obviously another program could be written, perhaps something like a desk calculator for manipulating one-dimensional signals. It could have a signal stack and use a reverse-polish notation.)

Drunkard's walk

With 13 nodes, there are $13 \times 13 - 13 = 156$ possible pairs, hence that many possible unique journeys. This presented an interesting programming problem for me—to avoid getting overloaded with details. What I did is to use random numbers to simulate a drunkard's walk through the directed graph. The walk begins at the signal that has just been edited. The walk ends at the other signal displayed on the screen. On a preliminary walk no calculations are done, only the path is observed in preparation for a second walk where the calculations will be done. On the preliminary walk, the drunkard keeps a list of the nodes he passes. If he encounters any node for a second time, the path from the first encounter of that node to its second encounter is crossed off his list. Thus when the ending node is eventually reached, his list is a path from the desired starting node to the desired ending node. This path is unique (by the design of the network). Lastly, this path is used for a second walk on which the computations are done.

Doing only what needs to be done

At each node, a time stamp is maintained for the signal. Whenever a signal is hand edited, then it is given a time stamp that is more recent than any other signal. Time stamps are updated by the programs that make the node-to-node transitions. When a node-to-node transition has a unique inverse, then the time stamp of the output is made equal to that of the input. When the inverse operation is nonunique, then the time stamp of the output is made greater than the input.

The result of all this time stamp maintenance is that some calculations may be avoided. When a node-to-node transition is called, it first compares the time stamp of the input and the output. If the output is later or equal the input, then the computation is omitted because recalculation would only regenerate the same thing. I found the calculations so fast that the time stamping was hardly worth bothering about.

Constraints

Many signal spaces have constraints. For example, some signals must be even while others are odd or causal. Spectra must be positive; reflection coefficients are bounded in magnitude by unity, etc. These constraints are forced again after any signal is edited, and they are tested and forced after any signal is computed. The signals occasionally disappear, an occurrence I associate with floating point overflow, and occasionally the program crashes.

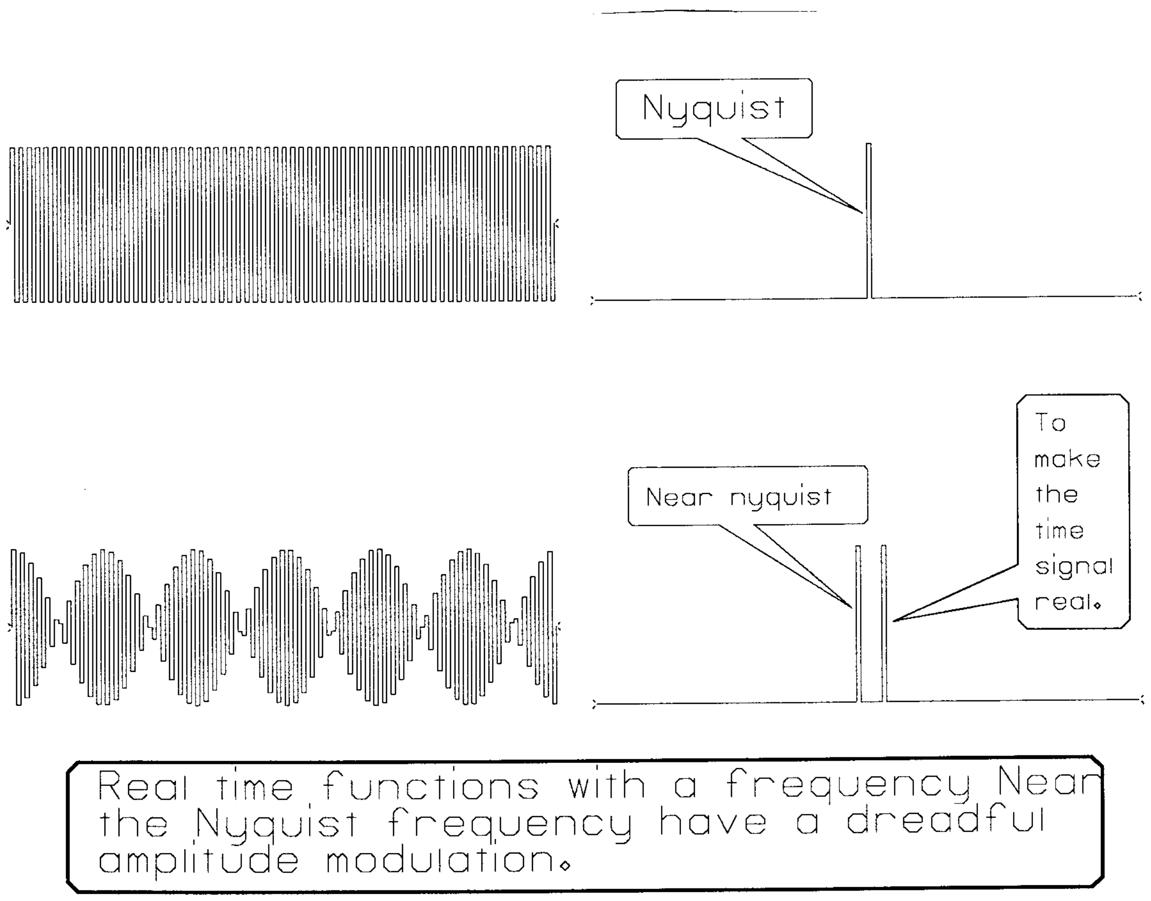
LEVEL OF EFFORT

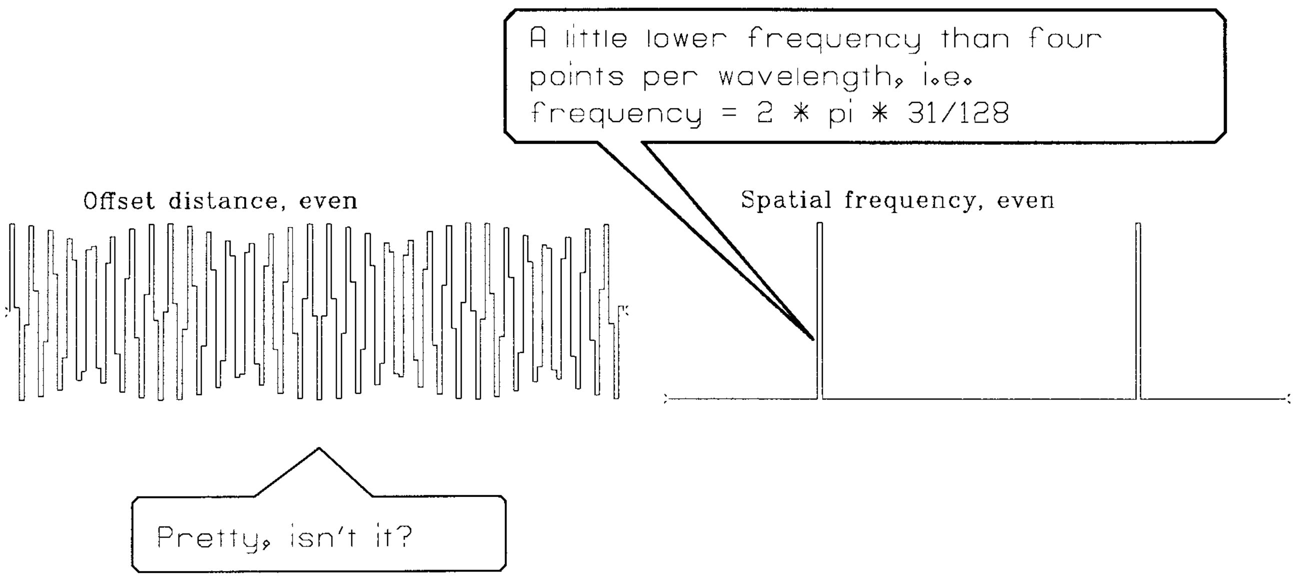
I estimate the level of effort at two and a half concentrated man-months. That comes to about three days for each control panel button, one day for conception, one day for implementation, and one day for debugging. I am pleased to report that no more than about ten days were spent in bewilderment pulling parts from a program that had no rational behavior.

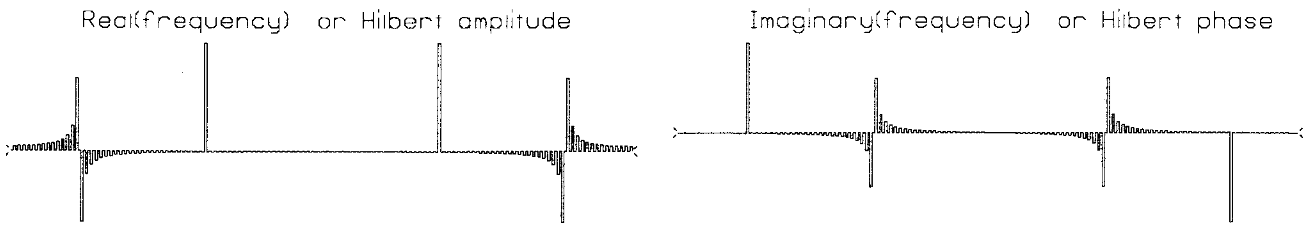
REGRETS

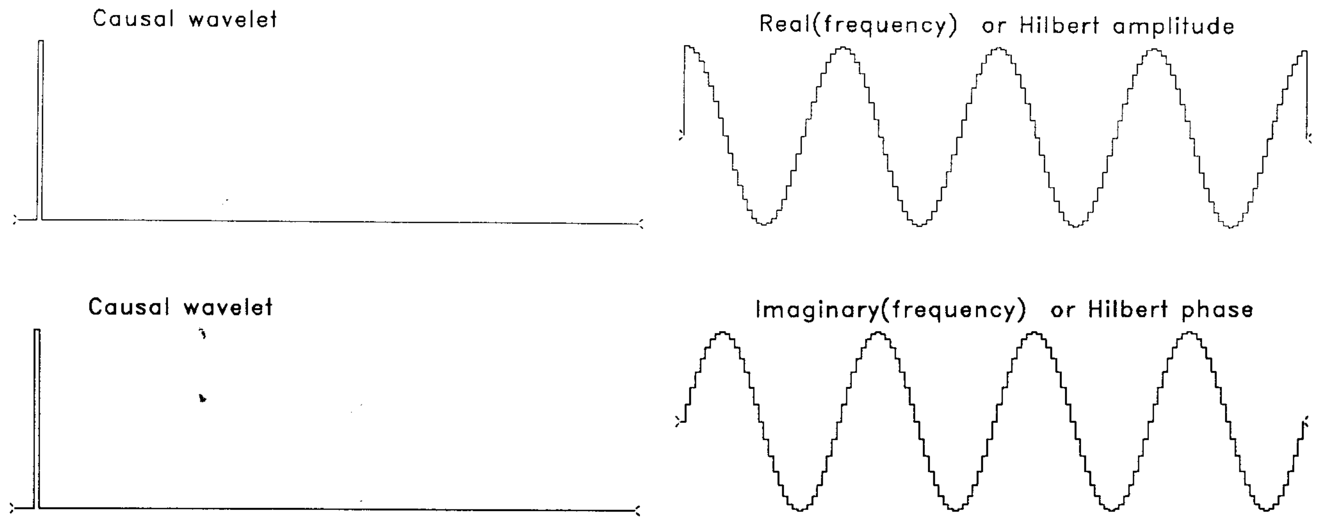
- This program could benefit a wide range of people. I regret that it runs only on an expensive workstation and not on an Apple or an IBM-PC. It requires little computer power.

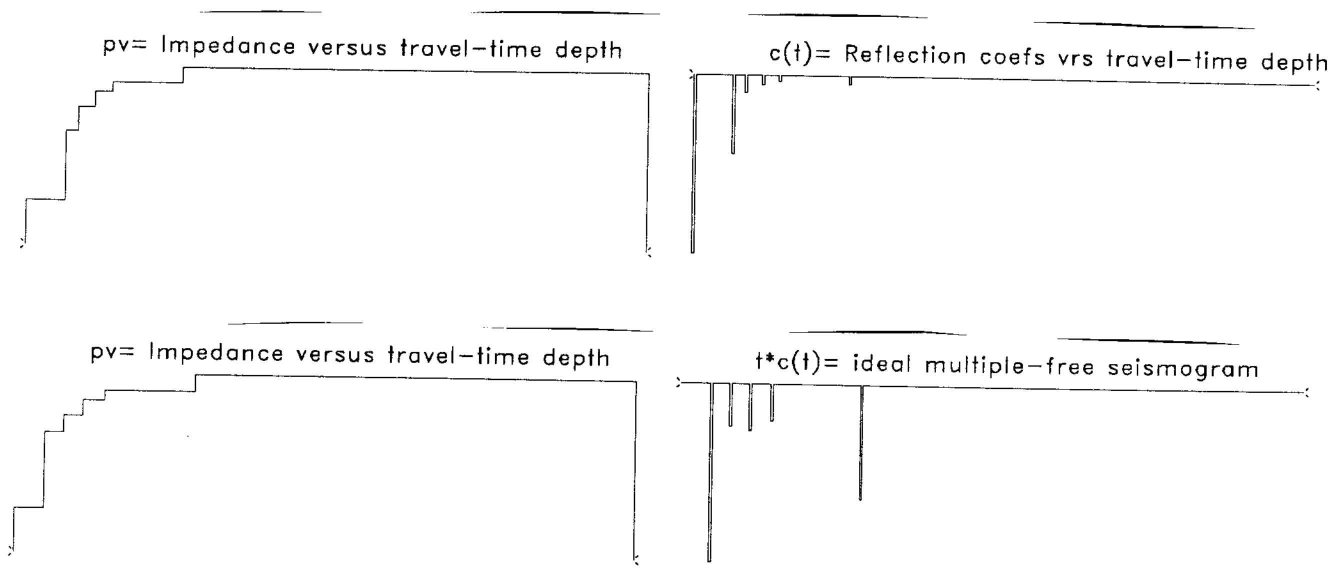
- Although this program does nearly everything it was intended to do, one-dimensional seismology is just not as much fun as two- or three-dimensional seismology.
- I never did meet one of my original goals—to learn how to make a one-dimensional signal editor that could be plugged into other programs as handily as a Fortran subroutine can be.

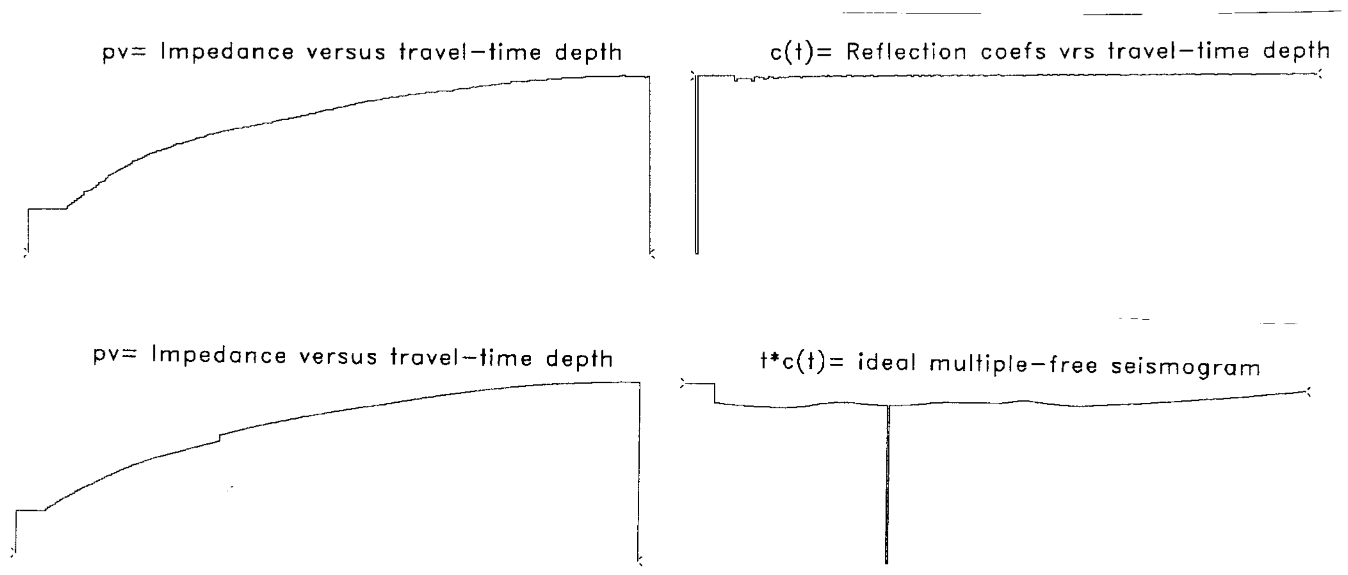


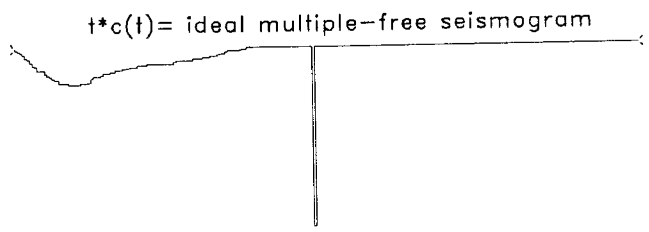

A little lower frequency than four points per wavelength, i.e. frequency = 2 * pi * 31/128
Offset distance, even
Spatial frequency, even
Pretty, isn't it?

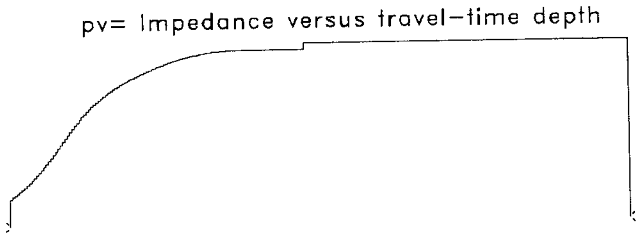

Real(frequency) or Hilbert amplitude
Imaginary(frequency) or Hilbert phase

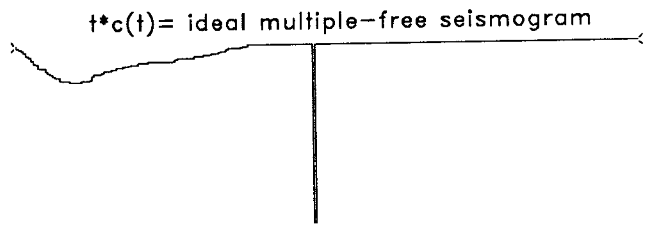

Causal wavelet
Real(frequency) or Hilbert amplitude
Causal wavelet
Imaginary(frequency) or Hilbert phase

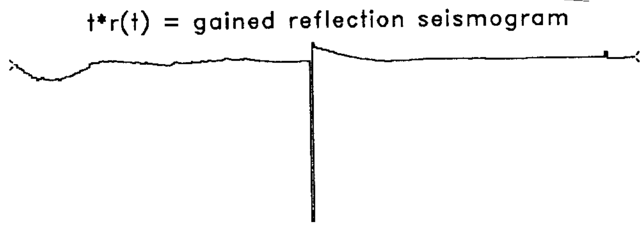

pv= Impedance versus travel-time depth
c(t)= Reflection coefs vrs travel-time depth
pv= Impedance versus travel-time depth
t*c(t)= ideal multiple-free seismogram

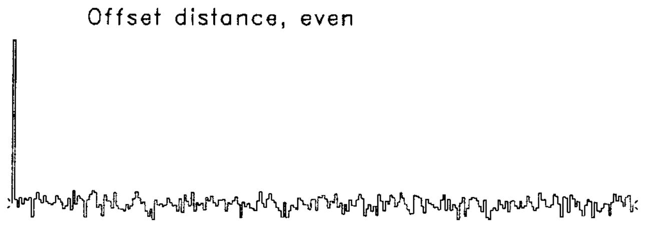

pv= Impedance versus travel-time depth
c(t)= Reflection coefs vrs travel-time depth
pv= Impedance versus travel-time depth
t*c(t)= ideal multiple-free seismogram

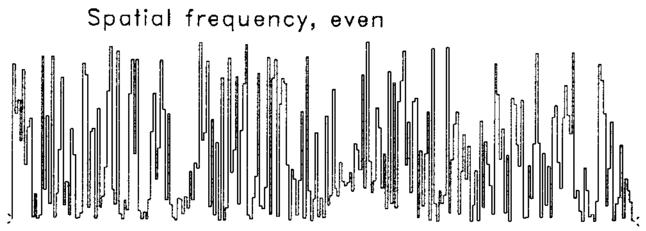

t*c(t)= ideal multiple-free seismogram

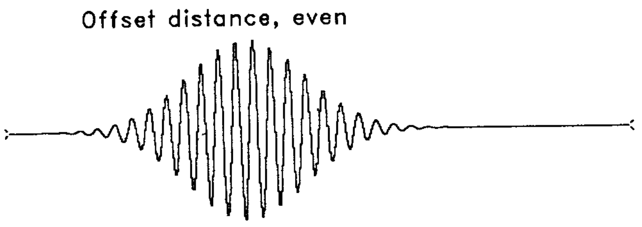

pv= Impedance versus travel-time depth

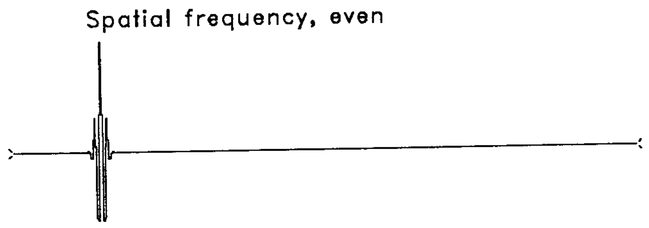

t*c(t)= ideal multiple-free seismogram

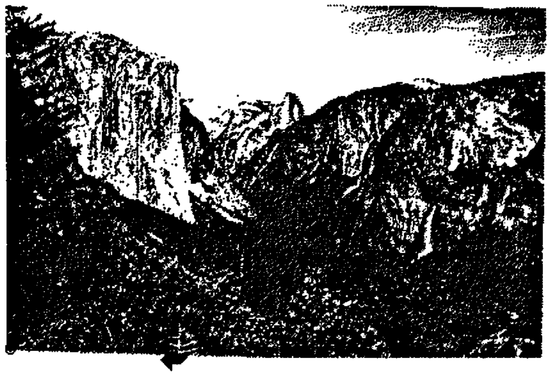

t*r(t) = gained reflection seismogram


Offset distance, even

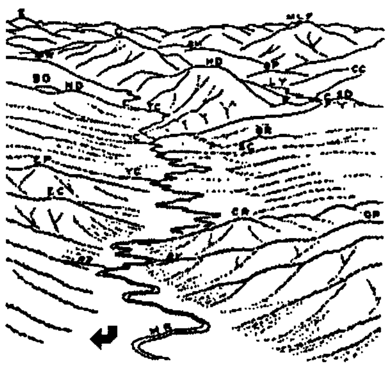

Spatial frequency, even


Offset distance, even

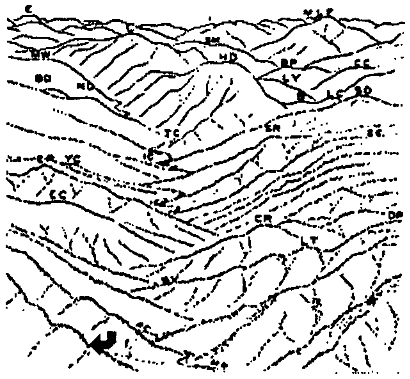

Spatial frequency, even



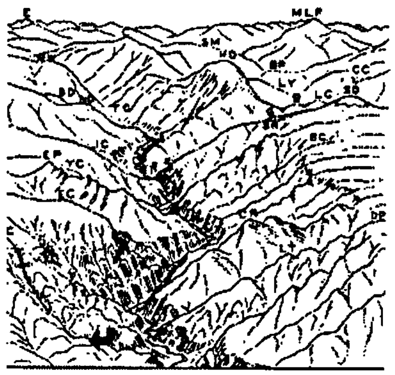
Broad-valley stage



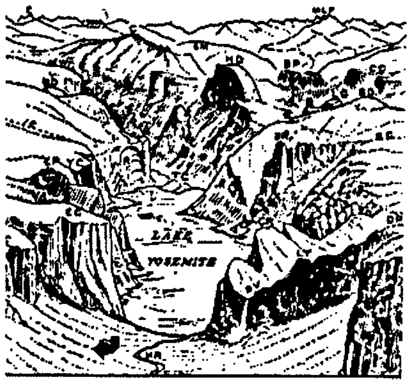
Mountain-valley stage



Canyon stage



Near present time


LAKE
YOSEMITE