Interactive filter design in the Z-plane

Jon Claerbout

ABSTRACT

I wrote a program for filtering seismic gathers or sections on a Sun 3/110 workstation. The user specifies the filters with a mouse by placing poles and zeros in a complex plane represented on a video screen. A display of the filter and its spectrum is updated "instantly" after any poles or zeros are moved. Filtering a plane of data takes about 3-30 seconds.

INTRODUCTION

Choice of a display filter is important for both field data and synthetic data. Goals for filter design that are expressed in the frequency domain generally conflict with other goals in the time domain. For example, when a filter is specified by frequency cutoffs and rolloffs, then the time-domain behavior, i.e. filter length, phase shift, and energy delay, are left to fall where they may.

Z-plane theory conveniently incorporates causality and relates time and frequency domains. This led me to write a program, the subject of this paper, in which filters are specified by using a pointer to place and move poles and zeros in the complex Z-plane. Since these root locations are related to analytic expressions for the filter and its spectrum, the updating of displays of both domains is virtually immediate (for moderate numbers of roots). Recursive filtering is fast but not immediate; filtering a family of seismic traces requires about 3-30 seconds on a Sun 3/110 workstation (which has a floating point chip, but not a floating point accelerator).

A program of this type should be useful to anyone designing filters for data display. Being interactive, you can easily experiment with a wide variety of parameters before making hard copies. I think a video screen is better for comparisons than hard copies anyway. Also, a video test takes only seconds whereas hard copy production takes minutes (or hours).

Hands-on theory

I was further motivated to write this program because of my experiences with students. Master's degree students often have difficulty grasping the material, and I suspect many of them forget it soon after the course is over. They want and need ***hands-on experience*** to supplement the abstract knowledge. PhD students grasp the abstractions but with little hands-on experience, a few years after taking the class they often fail to apply it. The most flagrant case, undoubtedly the one that prodded me into this action was an elaborate seismic modeling recently done here at SEP where the final display filter spoiled the result by introducing so much delay as to frustrate identification of simple multiple reflections.

Causality

Causality enters in seismology in two important ways: First, the filtering effect of reverberation and absorption in the weathered layer. Second, field recording itself is done in real time.

This is not to say there is no role for noncausal filters such as symmetric filters. Vibrator correlation functions are symmetric. In velocity analysis we should either compensate for filter delay or else use symmetric filters.

To accommodate both causal and phaseless filtering applications, the Z-plane filter program has a switch between causal operation, and operation where the filter is applied both directions in time. If your principal application is phaseless band pass filtering, then the Z-plane approach to filter specification seems inappropriate because a "causal bandpass filter" is a contradiction in terms—no causal filter can have a range of zero values along the frequency axis. This contradiction may partly explain the well known difficulties in tying seismic survey lines that cross.

THE DISPLAY

The program displays four planes: (1) an impulse response graph, (2) a frequency response graph, (3) a complex frequency plane for roots, and (4) a seismic data plane (such as a gather or section). Planes (1), (2), and (3) are line drawings or "vector plots" and they update immediately whereas plane (4) is a variable brightness plane and it updates only on command.

Seismic data plane

The seismic data plane is displayed as raster information, i.e. gray levels, with clipped values shown in a dull red. Like all reflection seismologists, I like to see the time axis running down. So I found it irksome that the hardware conspired to make me plot the seismograms with the time axis running horizontal. First, there are a limited number of pixels on a video screen and the time axis speaks loudest for the longer horizontal dimension. Second, a more complicated and compelling reason: Because the filtering takes a bit of time, I decided to speed the human/machine interaction by plotting each filtered trace as soon is it is ready, rather than asking the user to wait the 3-30 seconds or so until all traces are ready. So, the user touches the mouse on any trace and the filtering begins at that trace and continues surrounding the trace with more filtered traces until all (re)filtered traces are displayed (or until interrupted by an impatient operator who wants to change parameters). The way a trace is put on the screen is to prepare a brightness byte for each time point in the seismogram. Ordinarily in seismology we need every allowed pixel on the time axis (and then some) but we usually have pixels to spare on the space axis. So we almost always zoom horizontally, i.e. we replicate a trace and plot it again along side itself. The problem is that the Sun hardware function (pw_rop) used here is based on replicating horizontal scan lines, i.e. there is no means of replicating a vertical line along side itself. This is why traces are plotted horizontally.

The *clip* is an important display parameter. The program takes the clip as 1% more than the maximum value of the 30 time points centered on the cursor. So to replot the filtered data with a different clip, you touch the data in a different place.

There are no numbered axes on the data plane because none are needed. As the user moves the mouse across the data plane, the values of time and space are written near the ends of the axes. These values are more accurate than you could easily read from numbered axes.

Complex frequency plane

Likewise the location of the cursor in the complex frequency plane is printed beneath the plane as the cursor moves. I had a choice whether to display a complex Z plane or a complex ω plane. Since only positive frequency values are needed, the required Z plane would be a semicircle. I chose the Cartesian axes of the ω plane instead so that the real frequency axis used to display root locations would be the same axis used to display spectra.

The letters "z" and "p" are plotted in this plane to show the locations of poles and zeros. The location of these roots is under the exact center of the letter. You may put one letter exactly on top of another, but if you do, it will disguise the multiplicity of the root.

You should recall from Z-plane theory that in order to keep the filter response real, any pole or zero on the positive ω axis must have a twin on the negative ω axis. To save space, I don't plot the negative axis, so you don't see the twin. Thus you need to be careful to distinguish between a root exactly at zero frequency (or at Nyquist frequency) with no twin, and a root slightly away from zero (or Nyquist) with a nondisplayed twin.

Manual dexterity is not required to place poles or zeros exactly on the real ω axis. Let the complex frequency be decomposed into its real and imaginary parts, i.e. $\omega = \Re\omega + i\Im\omega$. All filters are required to be causal and minimum phase, i.e. all poles and zeros must be outside the unit circle in the Z-plane. Since $Z = e^{i\omega}$, the roots must all have negative values of $\Im\omega$, Any attempt to push a root to positive values of $\Im\omega$ simply leaves the root stranded on the axis of $\Im\omega = 0$. Likewise, roots can easily be placed along the edges $\Re\omega = 0$ and $\Re\omega = \pi$.

Although mathematics suggests plotting $\Im\omega$ along the vertical axis I found it more practical to plot something like the logarithm of $\Im\omega$ because you frequently need to put poles quite close to the real axis. The logarithm is not exactly what you want either because zeros may be exactly on the unit circle. I couldn't devise an ideal theory for scaling $\Im\omega$. After some experimentation, I settled on $\Im\omega = -(1 + y^3)/(1 - y^3)$ where y is the vertical position in a window of vertical range $0 < y < 1$.

Frequency response graph

A frequency response graph displays the amplitude spectra of the current filter. On the same axes, the amplitude spectrum of a portion of data can be displayed. The Burg method is used to compute the spectrum of the hundred time points of the trace centered on the cursor. Display is almost immediate after touching the "s" key.

Impulse response graph

The causal impulse response filter is replotted whenever the operator makes adjustments in the complex ω plane. Additionally I print the first several coefficients of the numerator and denominator polynomials. Besides their intrinsic interest, these time domain displays help alert the operator to root multiplicities.

COMMAND LIST

Any time you touch the seismic data plane (click the left mouse button on it) the data is refiltered and scaled to clip where you touched it. Touching the complex frequency plane causes the nearest pole or zero to jump to the cursor. You can also move roots by dragging (holding the left mouse button down while you move the mouse).

Functions activated by the keyboard are:

- T — Scale data by another power of $\sqrt{t}$.
- t — Scale data by another power of $1/\sqrt{t}$.
- p — Add a pole in the complex plane at the cursor location.
- z — Add a zero in the complex plane at the cursor location.
- P — Delete the pole that is nearest to the cursor.
- Z — Delete the zero that is nearest to the cursor.
- Ee — Expand or contract the time axis on the impulse response display.
- s — Display the Burg spectrum of 100 time points around the cursor.
- S — Erase the Burg spectrum (to unclutter the display).
- v — Snapshot vector plots (line drawings) into a file.
- d — Dump the plane of filtered data to the standard output data cube.
- B — Put the currently displayed, filtered data plane into buffer "B."
- b — Swap the display between buffer "B" and the filtered data.
- c — Set filtering to be in the causal direction only.
- C — Set filtering twice, once causally, once anticausally.

EXAMPLES

The following examples show the three vector drawn planes without their text and without the raster plane.

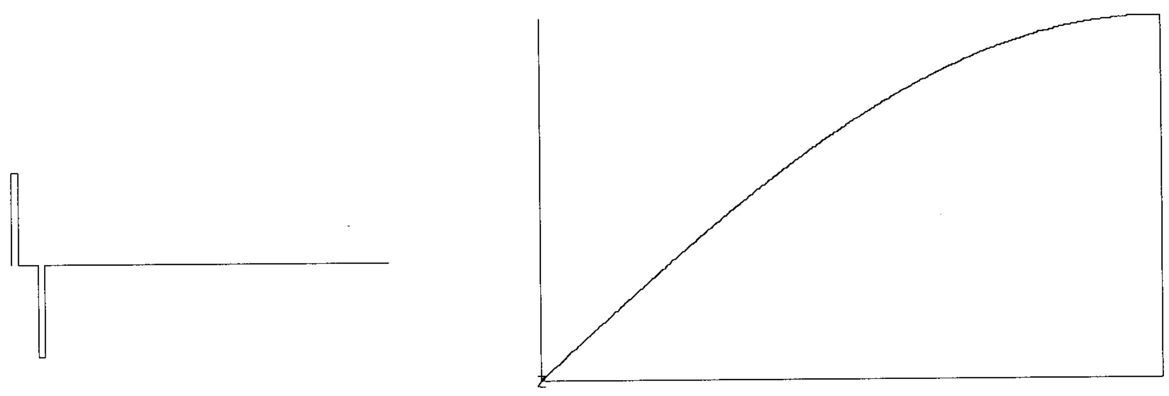

FIG. 1. A zero at $\omega = 0$ or $Z = 1$, is a discrete representation of the first derivative operator. It defines the approximation $\hat{\omega} = 2 \sin \frac{\omega}{2}$.

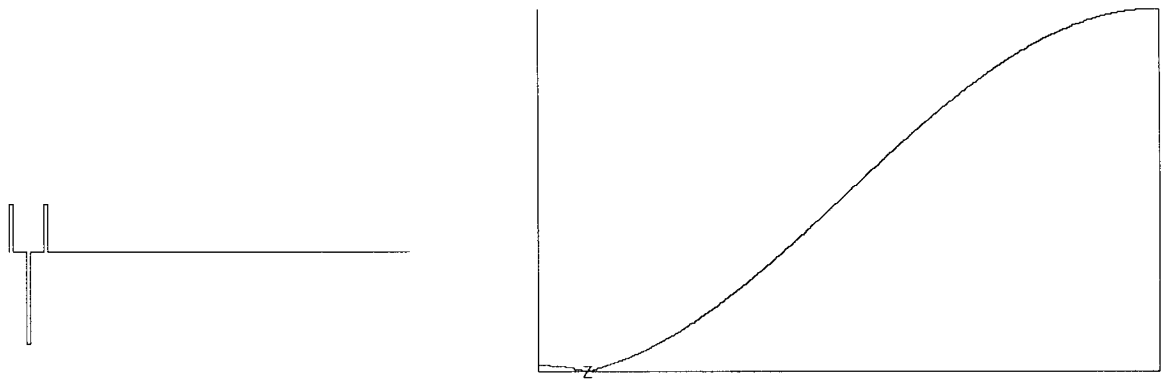

FIG. 2. When the zero is moved slightly off from $\omega = 0$ to a small positive frequency, then to keep the impulse response of the filter real, another zero is added at the negative frequency. It is nearly like having two roots at zero frequency and the time response is nearly (1,-2,1).

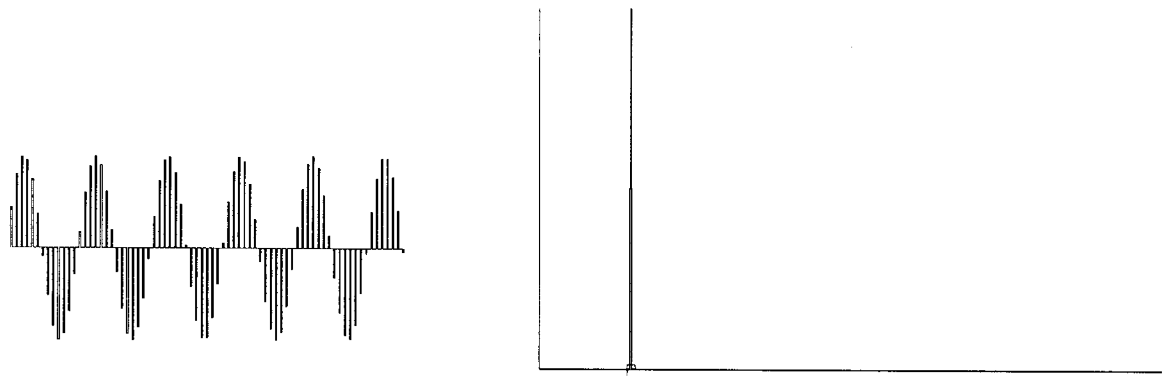

FIG. 3. A pole on the real axis gives a delta function at that frequency and a sinusoidal function in time.

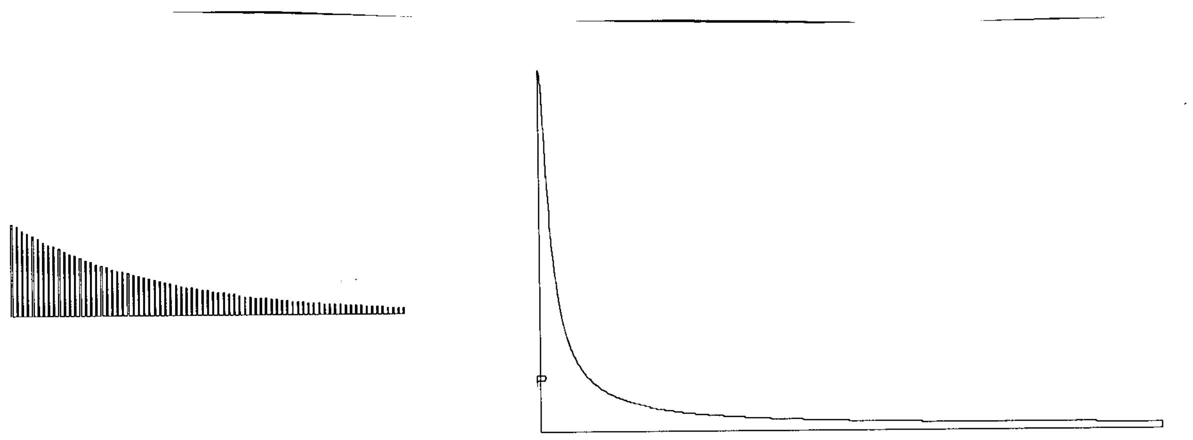

FIG. 4. A pole on the imaginary axis implies leaky integration.

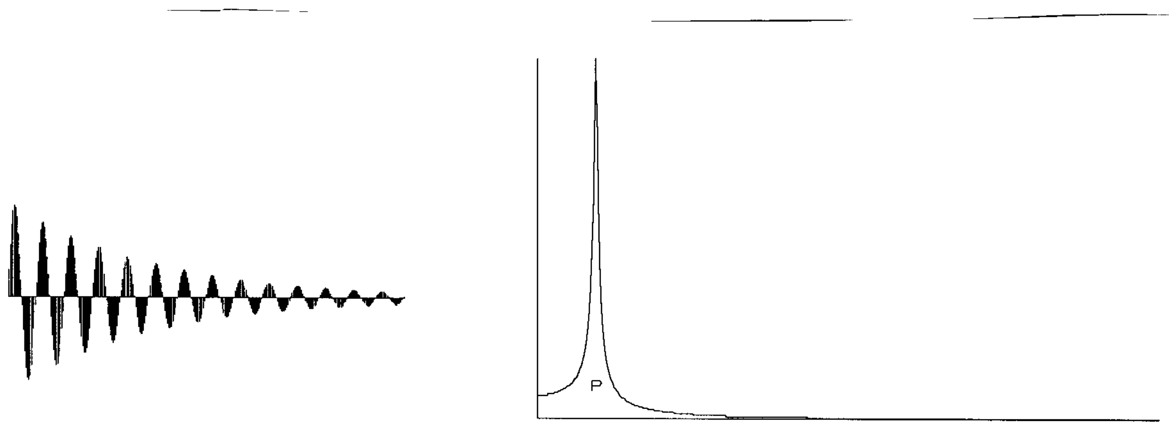

FIG. 5. A pole near the real axis gives a damped sinusoid in time.

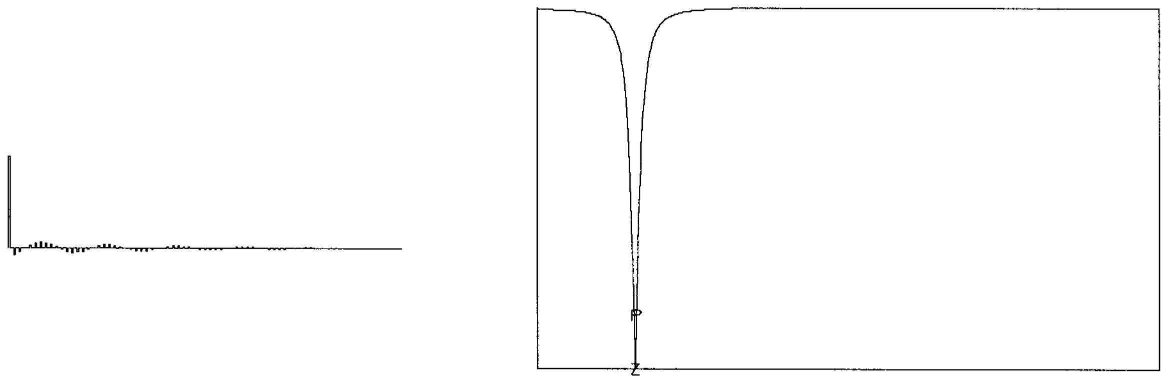

FIG. 6. A zero on the real frequency axis and a pole just above it gives a notch filter, i.e. the frequency is rejected while other frequencies are little changed.

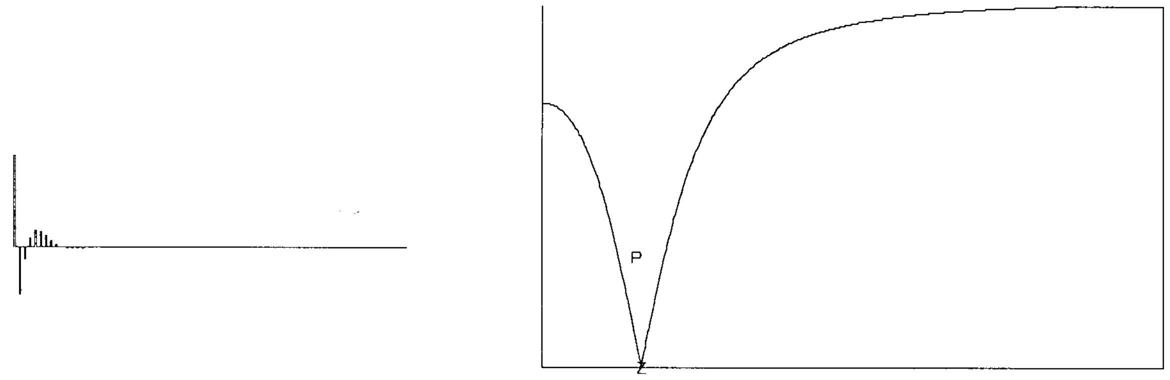

FIG. 7. A notch filter where the notch has been broadened by moving the pole further away from the zero.

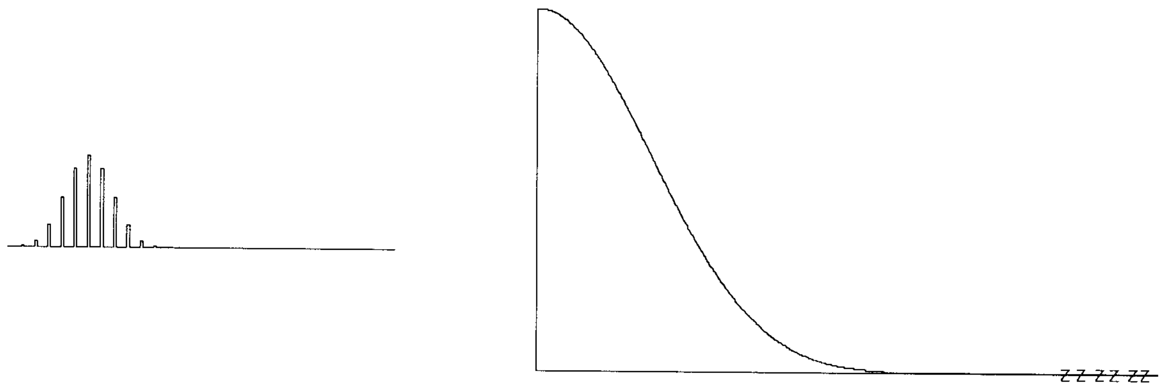

FIG. 8. Many zeros near the Nyquist. The coefficients of $(1 + Z)^N$ give Pascal's triangle which tends to Gaussian. Here minimum phase is a mathematical fancy. In reality, energy is long delayed after the onset time.

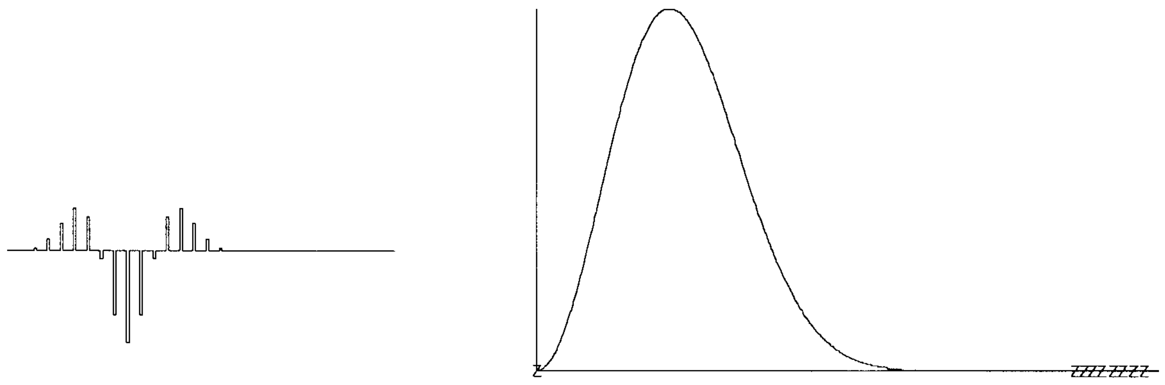

FIG. 9. Starting from the Gaussian and putting two more zeros at the origin gives us an old favorite wavelet, the second derivative of a Gaussian, also called a Ricker wavelet.

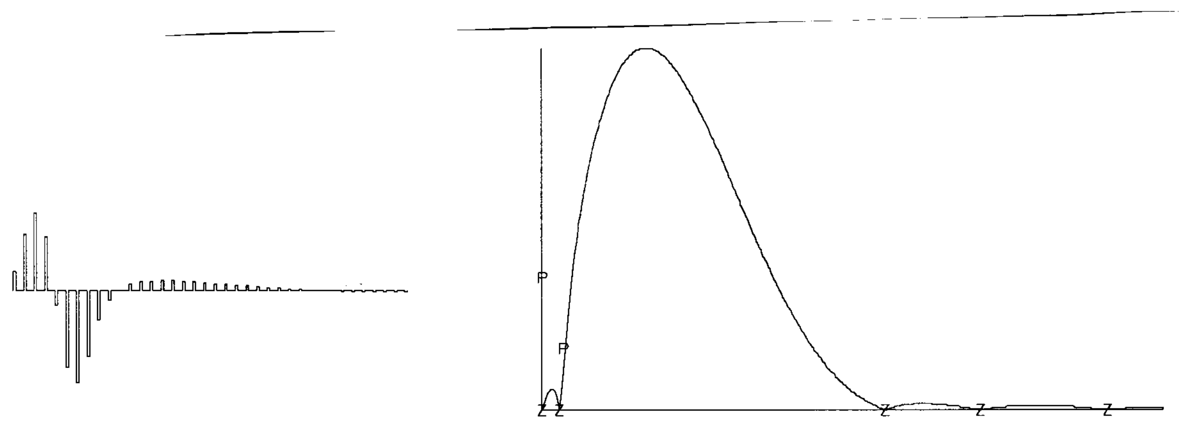

FIG. 10. My favorite wavelet for seismic modeling. I use some zeros at high frequency to force continuity in the time domain. I use a zero at the origin to suppress d.c. I like to simulate the suppression of low frequency ground roll so I put another zero not at the origin, but at a low frequency. Theory demands a conjugate pair for this zero so effectively there are three roots to suppress low frequencies. I used some poles to skew the passband towards low frequencies. They also remove some of the oscillation caused by the three zeros. (Each zero is like a derivative and causes another lobe in the wavelet). There is a trade off between having a long low frequency tail and having a rapid spectral rise just above the ground roll. The trade off is adjustable by repositioning the lower pole. The time domain wavelet shows its high frequencies first and its low frequencies only later. I like this wavelet better than a Ricker wavelet.

PROGRAMMING

The Z-plane program is written in a mixture of C-language and Fortran. Fortran (actually Ratfor) is used for all complex arithmetic. The graphics screen display and interaction is done with Sun Microsystem's proprietary software called *Sunview*. Data plane I/O is done with SEP's library "seplib". Vector plotting is done with a new local library "svplot" that merges some Sunview calls with some calls from "vplot", (SEP's older device independent library).

Level of effort

Much to my surprise and delight, the task took me only about two weeks. These two weeks, however, benefited from four months of Sunview learning and C refresher last summer when I wrote an interactive hyperbolic overlay program. Naturally I began from that program. After writing the Z-plane program I spent another week or two writing "svplot", a library that partially unifys SEP's old device independent vector plotting library with the interactive vector plotting calls from Sunview. Something like this had to be done to make the figures for this paper. Further delays resulted from bugs associated with a changeover of hard copy production from the Imagen to the Apple laser printer.

X-Windows

Rick Ottolini amazed and delighted me when he converted my Z-plane program into the X-windows system. This seemed to require only a day or two of his time. A useful thing about using X is that the computation need not be done in the same computer as the display. So Rick let the computations be done in the Convex minisuper computer while the display continues to be done on the Sun. The good thing about this arrangement is that the program doesn't seem to run slower when the filters have many poles and zeros.

Future versions of this program that are more ambitious in compute-power requirements should benefit even more from this arrangement. But our current version of X-windows is preliminary so I prefer to continue for now in Sunview.

BUGS & DEFICIENCIES

Theoretically, all the filters are minimum phase and polynomial division should never diverge. I find on rare occasions that the polynomial division is diverging. At this moment I am uncertain if there is a bug in the root-manipulation code or if there is a round-off problem. If it turns out to be a round-off problem, it will be readily solvable by dividing the roots one at a time from the data.

CONCLUSIONS

Like the hyperbolic overlay program, the Z-plane interactive filter specification is a useful tool that should soon be in every seismologist's "toolbox".

FUTURE WORK

I have long felt that spatial filters are underutilized. Most people understand the conceptual differences between dip filtering and spatial filtering but they have so little practical experience that they are unaware of opportunities or pitfalls. In IEI I teach causal dip filters. Such filters are as fast or faster than the time domain filters in this Z-plane program. We need an interactive program for spatial and dip filters.

Spatial filters and dip filters are not entirely cosmetic. For example, Fabio Rocca showed in some old SEP report that dip moveout before stack can be simulated by a filtering operation after stack. This concept was expounded again with great enthusiasm by G. Gardner at the 1987 EAEG meeting. I think an approximate interactive method is the best way to go.