Complex Cepstrum Processing of Digitized Transient Calibration Data for Removal of Echoes

L. B. Poché, Jr.

Methods Section
Underwater Sound Reference Division

September 30, 1977
### ABSTRACT (Continued on reverse side if necessary and identity by block number)

The complex cepstrum technique was investigated for possible use in removing echoes caused by early reflections when low-frequency, peaked-response transducers are calibrated with transient signals. The calibration environment permits a high signal-to-noise ratio and a free choice of input waveforms, both of which are advantages in complex cepstrum processing. The proper method of computation with this technique is discussed in detail. Cepstrum filtering methods are discussed. Synthetic measurement drops.
20. ABSTRACT (Continued)

Data produced by a J9 projector driven with a damped sinusoidal pulse and a pressure-release reflection are shown, and real measurement data from a low-frequency line array driven by a single-cycle sinusoid with complex surface and bottom reflections are also shown. In both the synthetic and the real data, the echoes were successfully removed using the complex cepstrum technique. Data were digitized directly and stored on digital magnetic tape. Oversampling was reduced by sifting to achieve the correct sampling rate required by the complex cepstrum method. The results indicate that echo removal performed by complex cepstrum processing can be accurate enough to have potential usefulness in calibration procedures.
# CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>CEPSTRUM PROGRAM THEORY</td>
<td>1</td>
</tr>
<tr>
<td>SAMPLING CONSIDERATIONS</td>
<td>6</td>
</tr>
<tr>
<td>EXPERIMENT</td>
<td>11</td>
</tr>
<tr>
<td>CONCLUSION</td>
<td>15</td>
</tr>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>15</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>16</td>
</tr>
<tr>
<td>APPENDIX A — Computer Programs and Subroutines</td>
<td>A1</td>
</tr>
</tbody>
</table>
COMPLEX CEPSTRUM PROCESSING OF DIGITIZED TRANSIENT CALIBRATION DATA FOR REMOVAL OF ECHOES

INTRODUCTION

Calibration procedures employing broadband transient signals have been under development for several years at the Underwater Sound Reference Division (USRD) of the Naval Research Laboratory. The principal advantages of the broadband transient system [1] over conventional continuous wave (CW) sweep-frequency or pulsed systems appear to be reduced calibration time and convenience in obtaining the complex receiving response of a hydrophone. Reflections from within the boundaries of the calibration medium interfere with observation of the received waveform, thus limiting the available processing time per pulse and hence the frequency resolution of the resulting calibration. In calibration of certain low-frequency, peaked-response transducers, this problem can impose extreme limitations. The transfer function of the calibration enclosure (which is taken here to include such bodies of water as lakes or pools as well as tanks) distorts the direct-arrival part of the signal through the process of convolution. It follows that the true direct-arrival part of the signal must be recovered by deconvolution. The calibration environment permits a high signal-to-noise ratio and a free choice of input waveforms. These two advantages greatly increase the prospects for success of the deconvolution method under study here.

Systems or classes of signal-processing techniques that obey a generalized principle of superposition have been called homomorphic. A well-known homomorphic system for deconvolution developed by Schafer [2] employs the complex cepstrum. Ulrych [3] gives an excellent summary of the theory of this system; however, his examples are somewhat confusing.

The basic steps necessary in the complex cepstrum process are given in Fig. 1. (Complex cepstrum is the name given to one function in the train of operations, arrived at in the bottom of Fig. 1. We shall refer to that function, and sometimes to the entire system, as the cepstrum.) These steps will be discussed in greater detail later. Although the required steps appear to be easily implemented, satisfactory results may be difficult to obtain until certain details of the basic steps are carefully studied. One purpose of this report is to discuss in detail the computation method of this system. A second purpose is to demonstrate the proper procedure for applying the process to real data.

CEPSTRUM PROGRAM THEORY

A computer program that will accomplish the deconvolution by means of the complex cepstrum technique can be written by performing the steps called for in Fig. 1.

Step 1 consists of simply transforming the time series into the frequency domain by performing a fast Fourier transform (FFT) operation.

Manuscript submitted June 7, 1977.
Step 2, taking the complex logarithm, is done by employing the relation

$$ \log z = \log r + i\theta $$

where $z = x + iy$. But $r$, the magnitude, equals $\sqrt{x^2 + y^2}$, and $\theta$, the phase, equals $\tan^{-1}(y/x)$. Log $r$ and $\theta$ become the real and imaginary components of a complex number representing the complex logarithm of $z$.

Steps 3 and 4 are called “unwrapping the phase.” They are extremely critical steps in the process. In order to be able to identify the periodic component representing the echo in this sequence, it is necessary to remove the numerous discontinuities in the phase that arise from the multivalued definition of $\tan^{-1}$. Each time the phase value reaches $\pm\pi$, a discontinuity of $2\pi$ appears in the function. These discontinuities must be forced out of the function by adding or subtracting $2\pi$ to all subsequent values, as appropriate. This process is performed in the computer program by comparing each successive point to its neighbor. If a difference of 3 or more is found between neighboring values, the second point is identified as a positive or negative discontinuity, as the case may be, and an adjustment is made to this and all succeeding points. The value 3 was arrived at empirically. The sampling used causes the actual values obtained between discontinuous points to vary. If the frequency intervals are too coarse, discontinuities can even be lost. This can be avoided by sampling for a long enough time to provide adequate frequency resolution. If sufficient time or record length is not available, zeroes can be added to the time sequence as required. Because the phase data are always antisymmetric, we can generate the second half of the data by forming an inverted mirror image of the first half. Then we need only to phase-unwrap the first half.
Removal of the discontinuities introduces a huge linear slope, which is removed next, in step 4.

Improper or incomplete removal of the linear phase will leave large components in the cepstrum sequence, which mask the true amplitude of the echo impulse train components. The method we use to remove the linear phase may be explained by letting $\delta T$ be the sampling interval and $N$ be the total number of samples. We wish to eliminate any net time delay. If there is a delay of $M$ samples, we wish to shift the data back by $M\delta T$. A time delay of $M\delta T$ shows up in the spectrum as a multiplicative factor of $\exp(iM\delta T\omega)$. In the phase of the spectrum (the imaginary part of the logarithm) it shows up as an additive term $\phi_d$, where $\phi_d = M\delta T\omega$.

Now

$$\omega = n\delta \omega \quad n = 0, 1, \ldots \left(\frac{N}{2}\right)$$

and

$$\delta \omega = \frac{2\omega_{\text{max}}}{N}$$

where

$$\omega_{\text{max}} = \frac{\pi}{\delta T}.$$

So

$$\phi_d = \frac{2\pi Mn}{N} \quad n = 0, 1, \ldots \left(\frac{N}{2} - 1\right).$$

In this case, $n$ ranges only to $(N/2 - 1)$ because the FFT requires that $N$ be an even number. The phase function will have a point of symmetry at $N/2$, which has a value of zero. The total phase is then

$$\phi(n) = \phi_0(n) + \phi_d(n) \quad n = 0, 1, \ldots \left(\frac{N}{2} - 1\right).$$

By the integral definition of $M$, its value can be chosen such that the absolute value of $\phi_0(N/2)$ is always less than $\pi$. If $N$ is large, $\phi(N/2)$ is approximately equal to $\phi_0(N/2 - 1)$. Then the total phase,

$$\phi \left(\frac{N}{2} - 1\right) \approx \phi_0 \left(\frac{N}{2}\right) + \phi_d \left(\frac{N}{2} - 1\right) \approx \phi_0 \left(\frac{N}{2}\right) + \frac{2\pi M (N/2 - 1)}{N} \approx \phi_0 \left(\frac{N}{2}\right) + \pi M - 2\pi \frac{M}{N},$$
Again, if $N$ is very large compared to $M$, the last term is negligible. We have seen that the second term must be less than 1. So $M$ is the integer closest to $\lfloor(N/2) - 1\rfloor$. To form a rounded value, set $M$ equal to the integer value

\[
\frac{\phi(N/2 - 1)}{\pi} \approx M + \frac{\phi_0(N/2)}{\pi} - 2 \frac{M}{N}
\]

Since $\phi_0(n) = \phi(n) - \phi_d(n)$, the corrected phase is given by

\[
\phi_0(n) = \phi(n) \left( \frac{\pi M}{N/2} \right) n.
\]

(The term in brackets is the correction increment, FAZINC, used in the computer program given in the Appendix.)

Step 5, which follows, consists of performing the inverse transform by an FFT operation. This yields the complex cepstrum (which is actually a real function with no imaginary part, because the real part of the log spectrum is symmetric and the imaginary part is antisymmetric).

Step 6, called linear filter, may apply to any operation that is intended to separate the echo impulse train from the underlying cepstrum curve. In our case we wish to remove the echo. The simplest method might be to zero all values beyond a chosen point. That operation is called an early-pass filter, corresponding to the low-pass filter in the frequency domain. Because the cepstrum tends to concentrate the direct-arrival signal in the earliest part of the curve, most of that signal would be undisturbed by such an operation. The process's effectiveness depends mostly on how well the direct-arrival and echo portions of the cepstrum can be separated. This in turn depends on how close in time the echo and direct signal are.

A more specific method of removing the echo would be to identify the individual points in the echo impulse train and set those to zero, leaving the intervening parts of the
direct-arrival cepstrum undisturbed. This operation is termed a comb filter. In practice, it allows removal of echo signals of shorter delay than the early-pass filter. But zeroing a sequence of points can produce a disturbance when the underlying cepstrum is non zero at those points. A better method would be to try to replace the sequence of points with values from an estimate of the echo-free cepstrum. In our observations of the cepstra of properly sampled, band-limited signals, the direct-arrival signal component seems to have an alternating-point structure which can be immediately visualized by referring to it as “Jaws.” The best linear approximation for this type of structure cannot be made from adjacent points, but it can be made directly from the points two samples removed from the central point. Thus, if an echo impulse point to be removed is located at point $K$, the straight-line approximation between $K - 2$ and $K + 2$ is used to replace $K$. The algorithm also works well for smooth curves. A general expression to designate values of $K$ is

$$K = M \times n + 1 \quad n = 1, 2, 3, \ldots, \left(\frac{N}{M}\right)$$

where $M$ is the delay interval between signal and echo in sample points and $N$ equals the total number of sample points in the sequence. A second pass may be formed by

$$K = M \times n + 1 - N \quad \left(\frac{N}{M}\right) < n < \left(\frac{2N}{M}\right)$$

In both cases $n$ is always an integer.

Steps 7 through 10 are performed to reconstruct the time-series waveform with the echo removed. Step 7 transforms the sequence back into the frequency domain with an FFT operation. Next, in step 8, the linear phase component is restored using the same increment that was determined in step 4. In step 9 complex exponentiation is performed. This returns the spectrum to a linear state.

Between steps 1 and 2, the first term in the real part of the spectrum (the DC term) was tested for algebraic sign. This information was stored, and if that sign was found to be negative, the sign of the entire spectrum was reversed. The original sign is now restored just before step 10. Schafer [21] gives a good discussion of the reason for this operation.

Step 10 is an inverse FFT operation to return the sequence to the time domain. We now have the reconstructed time-series signal with echo removed.

With a computer program to perform these steps in hand, we tried removing echoes from some experimental data. The results were generally unacceptable. In order to discover where our problems were, we decided to apply the program to some more easily controlled synthetic data.
SAMPLING CONSIDERATIONS

The first data sequence used to test the cepstrum program was a simulated waveform generated from an analytic function and forced to appear much like the waveform used by Ulrych (Fig. 2). The data sequence is 128 samples long. An echo of 0.9 magnitude and delayed by 12 samples was then added, making the composite signal appear as in Fig. 3. The total wavelet occupies about 45 samples. After the first FFT (step 2), one thing immediately apparent about the simulation was the very narrow band-width of the spectrum. This result may be seen in Fig. 4, where the right half represents the negative-frequency part of the transform translated upwards (this is the appearance of normal output from an FFT routine). Because the energy in the high-frequency part of the spectrum approaches zero, the logarithm of the magnitude produces large negative values, and the phase curve becomes erratic. After considerable experimenting, we were able to keep these erroneous values from swamp ing the higher energy parts of the spectrum by introducing a small impulse, or “glitch,” into the time series. Its echo was produced with the same amplitude and time delay as that of the data signal, the only intention being to supply some energy to the high-frequency parts of the spectrum. When the glitch is introduced, the real part of the complex logarithm appears as in Fig. 5. Note that the effects of the glitch and its echo are apparent in the high-frequency area. The results of phase-unwrapping are seen in Figs. 6 and 7. Figure 6 shows the imaginary part of the data. The discontinuities have been removed from the left half but not from the right. It was necessary to place the glitch at the center of the data impulse to avoid introducing into the imaginary part another phase component representing delay between the data impulse and the glitch. This phase component would appear as a different slope over the glitch-dominated frequencies in Fig. 6, producing a kink in the linear ramp. This double slope would prevent the linear phase from being minimized and cause masking of the impulse train in the cepstrum. The complete unwrapped phase signal appears in Fig. 7. There is a considerable difference in vertical scale from Fig. 6. The complex cepstrum is seen in Fig. 8.

The echo energy is represented by the alternating impulse train. The first spike in Fig. 8 is at the 13th sample, and succeeding spikes appear at each 12th sample following. The right half of Fig. 8 represents the same negative-frequency translation effect shown in Fig. 4, except that the diminishing impulse train does not belong in the negative-frequency region. Furthermore, if the vertical scale is changed to magnify the curve, it can be seen that the impulse train wraps around the cepstrum modulo 128.

The linear filter operation was performed using the special comb filter. The result of two passes of the filter, replacing every 12th point starting with 13 and then every 12th point starting with 5, is shown in Fig. 9. The result of the reconstructed time-series waveform is shown in Fig. 10.

We found no way to obtain the preceding result with this signal without resorting to the glitch while still maintaining the number of samples used. However, the problem posed by near-zero energy in the high-frequency part of the spectrum shown in Fig. 4 may be dealt with in a more satisfactory way. By sampling at or slightly below the Nyquist rate, we caused the important energy region to just fill the spectrum. This results in a very nice-looking cepstrum which, unfortunately, has an almost unrecognizable time-series waveform. The method is workable, however, and is the method used in the remaining examples.
Fig. 2 — Simulated waveform (128 samples)

Fig. 3 — Simulated waveform plus echo
Echo amplitude = 0.9
delay = 12 samples

Fig. 4 — Spectrum of simulated waveform of Fig. 3

Fig. 5 — Real part complex logarithm of simulated waveform plus "glitch"

Fig. 6 — Imaginary part corresponding to Fig. 5. Left half: after removal of phase discontinuities, right half: before removal of phase discontinuities.

Fig. 7 — "Phase unwrapped" imaginary part

7
Moreover, as will be discussed later, an oversampled waveform can always be generated from a properly sampled one.

Having successfully, if somewhat deviously, replicated the waveforms of Ulrych, we began to search for a better choice of driving signal. The problem was obviously with the sampling rate; and the solution was not to put a glitch in the signal, but to sample at exactly the right rate. One trial signal that seemed promising was the impulse response of an eight-pole Chebyshev filter. The idea here was to have a sharp high-frequency cutoff to avoid aliasing problems when the signal was sampled at exactly the correct rate. Figure 11 shows the Chebyshev filter impulse response as it would appear in an esthetically pleasing but oversampled form. In Fig. 12 we see the log spectrum derived from the oversampled signal. Note the sharp high-frequency cutoff and the undesirable low-amplitude, high-frequency components. Figure 13 shows the same waveform when the sampling rate has been changed to the proper value for cepstrum processing; Fig. 14 is the log spectrum, now showing only about 7-dB amplitude variation instead of 60 dB. Hardly any aliasing can be seen. In Fig. 15 an echo has been added to the waveform. This results in
a cepstrum (Fig. 16) which contains a very prominent sequence of single points representing the echo. This sequence of points was removed by using the special comb filter, and the reconstructed signal is shown in Fig. 17. A careful comparison of Fig. 17 and Fig. 13 shows that the two are virtually identical.

When employing artificially generated signals as input sequences in this manner, there is a normal tendency to form the time-series waveform by adding to the original signal a reduced-amplitude replica delayed by some integral number of sample points. In actual practice, the echo will seldom be delayed by an exact number of samples. The peak of the echo impulse might just as well fall midway between two sample points. In order to investigate the effect of this placement of echoes on the cepstrum, we included in the original synthesizing program a provision for making the echo delay continuously variable between sample points. Figure 18 shows the cepstrum that results when the echo is introduced midway between sample points. Comparing this figure with the integral sample delay version (Fig. 16), we see that the impulse train peaks have been reduced in amplitude and spread so that they are no longer single points. An attempt to remove the echo represented in Fig. 18 by the comb filter is generally unsuccessful, even with repeated passes, as can be seen from Fig. 19.

At this point it is clear that there are two rules pertaining to sampling that must be observed. First, the signal must be sampled at exactly the proper rate: a rate low enough that the spectrum does not extend beyond the frequencies of interest, and high enough that no significant aliasing takes place. Second, the samples must be chosen so that the echo falls on or very nearly on a sample point. The technique we used to accomplish this was to sample at as high a rate as possible consistent with the other requirements of the system (for example, storage). To obtain the desired sampling rate, we would sift the data, saving every \( P \)th point, where \( P \) represents the divisor necessary to give proper sampling, and then shift the starting point of the sequence until the echo fell as close as possible to a sample point. With this technique we were able to obtain good results from our records of real experimental data.
Fig. 13 — Properly sampled Chebyshev filter impulse response without echo

Fig. 14 — Log spectrum of properly sampled waveform

Fig. 15 — Waveform with echo added

Fig. 16 — Complex cepstrum of signal plus echo

Fig. 17 — Reconstructed waveform with echo removed

Fig. 18 — Cepstrum of signal with echo delayed midway between samples
EXPERIMENT

In order to test the program with the real experimental data, we used two types of signals that had been recorded previously. One type of waveform was obtained by using a damped sinusoid as the driving current, and the other type by using a single-cycle sine wave. The first was produced by driving a J9 transducer with a damped sinusoidal waveform and detecting with an H52 hydrophone. A reflector, consisting of a large, square piece of closed-cell neoprene cemented to a 1/4-in.-thick plexiglass plate, had been positioned so as to supply an echo with adjustable delay. The hydrophone signal was directly digitized using a Nicolet digital oscilloscope and recorded on a digital tape unit. The digitizing rate was set at the highest value that would allow the entire signal waveform to be stored within the 4096-point capacity of the oscilloscope. Results of two of these experiments are shown in Figs. 20 through 27. The original impulses with no echoes are reproduced graphically in Figs. 20 and 24. These waveforms were obtained by removing the reflector plate from the water. Next, the reflector was lowered to contribute the echo (Figs. 21 and 25). The driving waveform in the first four figures (Figs. 20-23) was a damped, ringing impulse from an RLC circuit. In the second set of figures (Figs. 24-27), the circuit was adjusted to give a much longer time constant, so that only the first half-cycle is seen here. The early part of the impulse is due to the transient response of the J9 transducer. Figures 22 and 26 show the cepstrum resulting from each waveform. Note that in the case of a pressure-release echo, the echo appears in the cepstrum as a sequence of negative impulses only, instead of as an alternating sequence. The same comb filter that was used previously has been applied; the reconstructed results are shown in Figs. 23 and 27.

Another type of data used was a sample of actual transient calibration data from an element of a line array hydrophone. The data were recorded at our Leesburg Facility. Rigging is usually done there at a depth where the pressure-release surface reflection will arrive at approximately the same time as the hard reflection from the side walls of the
spring. In the case of omnidirectional instruments this results in a minimum amplitude but somewhat distorted echo. Figures 28 and 29 show the received waveform in the oversampled and properly sampled forms, respectively. Bearing in mind the distorted form of the echo, we were not surprised to see the clearly unspectacular cepstrum as it appears in Fig. 30. What is surprising is the degree to which the echo is eliminated with a single pass of the comb filter (Fig. 31).

Fig. 20 — Experimental waveform from J9 transducer

Fig. 21 — Experimental waveform with echo

Fig. 22 — Complex cepstrum of signal plus echo

Fig. 23 — Reconstructed waveform with echo removed
Fig. 24 — Experimental waveform from J9 transducer

Fig. 25 — Experimental waveform with echo

Fig. 26 — Complex cepstrum of signal plus echo

Fig. 27 — Reconstructed waveform with echo removed

Fig. 28 — Transient calibration signal with echo, oversampled

Fig. 29 — Signal plus echo, properly sampled
In all of the previous examples, the properly sampled input signals and the reconstructed versions have a spiky appearance, which makes them difficult to relate to the oversampled data we are accustomed to seeing. It is desirable to restore the smoothed appearance to the sampled signal by interpolation. The interpolation formula used was derived from the expression:

\[
x(t) = \sum_{n = -\infty}^{\infty} x(nT) \frac{\sin 2\pi f_c (t - nT)}{2\pi f_c (t - nT)}
\]

which assumes that the signal \( x(t) \) is band-limited and the spectrum is zero for \( |f| > f_c \) and \( T = 1/(2 f_c) \). If interpolation points are chosen midway between sample points, the computation is reduced to

\[
x[(m + 1/2)T] = \sum_{n = 0}^{N} x(nT) \frac{(-1)^{m + n}}{\pi(m + 1/2 - n)}.
\]

The process of interpolating midpoints is repeated until the entire array of 512 points is filled. Figures 32 and 33 illustrate the results obtained using the modified program on the data of Figs. 29 and 31, respectively.

Up to now we have only briefly mentioned other obvious methods of linear filtering for the cepstrum (e.g., “late-pass” and “early-pass” filtering). These are analogous to high-pass and low-pass in the frequency domain. In our case, a high signal-to-noise ratio and the ability to select an optimum driving signal make it possible to keep the cepstrum “clean” enough for the echo impulse train to be identified clearly. When the source signal and the echo are closely spaced in time, the signal portion of the cepstrum and the

---

Fig. 30 — Complex cepstrum of transient calibration signal

Fig. 31 — Reconstructed signal with echo removed
impulse train may not be separated enough to avoid producing severe distortion in the source signal when early-pass filtering is used. Stated another way, it appears that the comb filter is able to remove echoes that have a shorter delay time than those the early-pass filter removes. Early-pass filtering can be useful in cases where the echo does not appear as a sequence of single points, either because of distortion or because it appears between sample points, as in Fig. 18. Late-pass filtering is of limited usefulness because the result of reconstruction indicates the time delay, which should be detectable in the cepstrum as well.

Fig. 32 — Transient calibration data with echo, properly sampled, smoothed
Fig. 33 — Reconstructed signal, echo removed, smoothed

Exponential weighting, as discussed in Ulrych [3] and elsewhere, has not been mentioned in the above discussion. The cepstrum method does not work at all if the echo sequence is not minimum phase (see Schafer [2]). Exponential weighting can be useful in forcing the echo components to be minimum phase, should there be multiple echoes. Too much weighting can cause considerable inaccuracies in recovering the source signal, however. The effect of exponential weighting is chiefly of use when it is more important to detect the delay time than to recover the source signal.

CONCLUSION

We have demonstrated that echo removal performed by complex cepstrum processing can be accurate enough to have potential usefulness in calibration procedures.

ACKNOWLEDGMENTS

The author gratefully acknowledges the assistance of Peter H. Rogers, A. Z. Robinson, and J. D. George. This work was performed under contract with the Naval Electronics Systems Command.
REFERENCES


Appendix A
COMPUTER PROGRAMS AND SUBROUTINES

This section contains listings of the computer program and subroutines pertinent to this report. The source language is Fortran IV, and all programs were implemented on a Digital Equipment PDP-11/45 computer with a multi-user operating system. Subroutines that have not been listed either are specific to the operational details of our system and, as such, would be of little use to others, or are ordinary utility subroutines that probably already exist in a prospective user's library. In those cases, a full explanation of the functional nature of each subroutine is given.
**FORTRAN IV-PLUS V62-94**

L. B. POCHÉ, JR.

16:43:15 19-JAN-77  PAGE 1

---PROGRAM PPTES(3) 3 JAN 77

---THIS VERSION USES SINGLE PRECISION FOR

*** EVERYTHING ***

USES NO COMPLEX FUNCTIONS

0001 DIMENSION IHEAD(3), IRAY(200), JRAY(200), FAKIT(1024)
0002 BYTE ILFA(30), FILT
0003 PI=*3.14159265
0004 THIS CALL SUPPRESSES FLOATING ZERO DIV ERROR*
0005 CALL ERRSET(73, .TRUE., .FALSE., .FALSE., .FALSE., .FALSE.,)
0006 C OPTIONAL SUBROUTINE SUBSTITUTES ANALYTICALLY GENERATED
0007 C SIMULATED WAVEFORM (AFTER ULRYCH) FOR REAL DATA.
0008 WRITE(6,462)
0009 READ(5,463) IULRY
0010 IF(IULRY.EQ.1)GO TO 12
0011 IF(IULRY.EQ.1)GO TO 3
0012 C SPECIFY INPUT FILE*
0013 WRITE(6,20)
0014 READ(5,30) ICNT, ILFA
0015 ILFA(ICNT+1)=0
0016 C SET UP FILENAME ARRAY FOR INPUT FILE, SPECIFY # OF PTS
0017 C TO READ FROM FILE.
0018 WRITE(6,142)
0019 READ(5,109) NF
0020 C SPECIFY NO OF DATA ARRAY POINTS (UP TO 512). MUST BE
0021 C A POWER OF 2.
0022 WRITE(6,106)
0023 READ(5,109) N
0024 N=FLOAT(N)
0025 N=FLOAT(N)
0026 C CHOOSE COMB. EARLY PASS OR LATE PASS CATING TO BE DONE
0027 C ON CEPSTRUM.
0028 WRITE(6,11)
0029 READ(5,34) FILT
0030 IF(IULRY.EQ.1)CALL UF(FAKIT)
0031 IF(IULRY.EQ.1)GO TO 3
0032 C CALL THE INPUT SIGNAL FILE-READING SUBROUTINE. THIS SUBR.
0033 C ALLOWS VARIABLE STARTING POINT & SIFTING FACTOR,
0034 C LOADS DATA IN ARRAY.
0035 CALL FILRED(FAKIT, ILFA, NF)
0036 C SELECT OPTION TO BYPASS PROGRAM. GOES TO DIRECT
0037 C INTERPOLATION OF SIFTED INPUT DATA
0038 WRITE(6,230)
0039 READ(3,463) IORIC
0040 IF(IORIC.EQ.1)GO TO 230
0041 462 FORMAT(’8 IF YOU WANT ULRYCH TYPE 1: ’)
0042 463 FORMAT(150)
0043 C OUTPUT A FILE FOR PLOTTING THE RAW DATA

A2
C OPTION TO USE EXPONENTIAL WEIGHTING
C IF EXP. WEIGHTING IS USED, SPECIFY EXP., IF NOT, INPUT 1

0030 WRITE(6,119)  
0031 READ(5,112) GAM  
0032 IF(GAM.EQ.1.E0)GO TO 21  
0033 FAN=FAM*CAM  
0034 DO 1 I=1,1HN,2  
0035 FAKIT(I)=FAKIT(I)*FAM  
0036 1 CONTINUE  
0037 10 FORMAT(2F9.4)  
0038 C TRANSFORM TO FREQUENCY DOMAIN  
0039 CALL FFTDP(FAKIT,N,-1)  
0040 C TRAP TO INVERT SPECTRUM IF DC COMPONENT IS NEGATIVE.  
0041 DO 350 I=1,1HN  
0042 FAKIT(I)=FAKIT(I)*FAKSIG  
0043 350 CONTINUE  
0044 C OUTPUT A FILE FOR PLOTTING THE SPECTRUM DATA.  
0045 CALL PLTFIL(FAKIT,0,N,1)  
0046 C TAKE COMPLEX LOG OF SPECTRUM-THE HARD WAY.  
0047 DO 54 I=1,N+2,2  
0048 FADIF=FAKIT(I)-FAKIT(I-2)  
0049 IF(ABS(FADIF)<.3.E0)GO TO 305  
0050 NR=NR+1  
0051 JRAY(NR)=I  
0052 GO TO 51  
0053 MP=NR+1  
0054 51 CONTINUE  
0055 C UNWRAP THE PHASE  
0056 C LOOK FOR DISCONTINUITIES  
0057 C AND COUNT POS & NEG ONES SEPARATELY  
0058 C START @ IMAG-PART OF 2ND PT, COMPARE IT TO 1ST PT. GO  
0059 C TO MIDPT., IF DIFFERENCE IS =>3, IT'S A DISCONTINUITY  
0060 DO 51 I=4,N+2,2  
0061 IF(ABS(FADIF)<.3.E0)51,41,41  
0062 IF(FADIF)42,999,43  
0063 NR=NR+1  
0064 JRAY(NR)=I  
0065 GO TO 51  
0066 MP=NR+1  
0067 51 CONTINUE  
0068 TDELT=0.E0  
0069 IF(NR.EQ.0)GO TO 305  
0070 C UNWRAP THE POS DISCS  
0071 DO 301 I=1,N  
0072 PDELT=PDELT+TDELT+2.E0*PI  
0073 301 CONTINUE  
0074 C
DO 302 J=IRAY(1),IRAY(1+1)-2,2
FAKIT(J)=FAKIT(J)-PDELT
CONTINUE
IF(NP.EQ.0)GO TO 306
C UNWRAP THE NEG DISCS
DO 303 I=1,NP
TDELT=TDELT+2.E0*P I
CONTINUE
DO 304 J=JRAY(I),JRAY(I+1)-2,2
FAKIT(J)=FAKIT(J)+TDELT
CONTINUE
CALL PLTFIL(FAKIT,0,N,1)
C OUTPUT A FILE FOR PLOTTING OF UNWRAPPED PHASE-1ST HALF
DO 305 I=4,N,2
FAKIT(I)=FAKIT(I)+FAZINC*FI
CONTINUE
CALL PLTFIL(FAKIT,0,N,1)
C OUTPUT A FILE FOR PLOTTING OF COMPLETE DOCTORED-UP PHASE DATA.
CALL PLTFIL(FAKIT,0,N,1)
C INVERSE TRANSFORM TO CEPSTRUM DOMAIN
CALL FFTDP(FAKIT,N)
C OUTPUT A FILE FOR PLOTTING OF THE RAW CEPSTRUM DATA.
CALL PLTFIL(FAKIT,0,N,1)
C PERFORM THE SELECTED GATING OPERATION.
IF(FILT.EQ.'A')GO TO 130
IF(FILT.EQ.'B')GO TO 140
IF(FILT.EQ.'C')GO TO 150
CALL COMBO(IHN,FAKIT)
GO TO 160
CALL LOWCD(IHN,FAKIT)
GO TO 160
CALL HIYCD(IHN,FAKIT)
GO TO 160
CALL PLTFIL(FAKIT,0,N,1)
C OUTPUT A FILE FOR PLOTTING OF THE MODIFIED CEPSTRUM.
C SCALE THE TRANSFORMED DATA.
DO 12 I=1,IHN,2
FAKIT(I)=FAKIT(I)/SN
CONTINUE
DO 132 I=2,IHN,2
FAKIT(I)=0.E0
CONTINUE
C TRANSFORM BACK TO LOG SPECTRUM (FREQUENCY)
FORTRAN IV-PLUS

CALL FFTDP(FAKIT,N,-1)

RESTORE THE LINEAR PHASE COMPONENT—1ST HALF OF DATA

DO 56 I=4,N,2
 F:FLOAT(I/2-1)
 FAKIT(I):=FAKIT(I)—FAZINC*FI
 56 CONTINUE

SYNTHESIZE THE 2ND HALF OF IMAG.DATA

FAKIT(N+2)=0.0
 DO 166 I=4,N,2
 FAKIT(IN+4—I)=—FAKIT(I)
 166 CONTINUE

EXPONENTIATE PHASE(ANTILOG)—THE HARD WAY.

DO 53 I=1,IUN,2
 DOLC:EXP(FAKIT(I))
 DA1IC COS(FAKIT(I+1))
 DBAS=SIN(FAKIT(I+1))
 FAKIT(I)=DOLC*DA1IC
 FAKIT(I+1)=DOLC*DBAS
 53 CONTINUE

REPORT THE # OF POS. & NEG. DISCONTINUITIES REMOVED IN
 UNWRAPPING

WRITE (6,109) NP,MP
 DO 360 I=1,IUN
 FAKIT(I)=FAKIT(I)*FAKINC
 360 CONTINUE

TRANS BACK TO TIME DATA

CALL FFTDP(FAKIT,N,1)
 DO 220 I=1,IUN,2
 FAKIT(I)=FAKIT(I)/SN
 220 CONTINUE

REMOVE THE EXPONENTIAL WEIGHTING, IF ANY. THIS GETS
 STICKY IN THE UPPER END OF ARRAY.

FAM=1.00/CAM
 IF(CAM.EQ.1.00) GO TO 250
 DO 2 1=1,IUN,2
 FAM=FAM*CAM
 FAKIT(I)=FAKIT(I)/FAM
 2 CONTINUE

SMOOTH THE UNDERSAMPLED DATA WITH INTERPOLATION. THIS
 OPERATION PLACES INTERPOLATED POINTS MIDWAY BETWEEN
 DATA POINTS. REPEATS THIS OPERATION UNTIL MAX ARRAY(512)
 SIZE IS REACHED.

NTRP=SQRT(FLOAT(512/N))
 DO 500 NIT=1,NTRP
 DO 510 MUL=1,N—1
 FKSUM=0.
 500 SYGN=(-1)**MUL/P1
 ID=—1
 CUL=1+0.5
 DO 520 I=1,N
 SYGN=—SYGN
 ID=1+2
 FKSUM=FKSUM+FAKIT(ID)*SYGN/(CUL—1)
 520 CONTINUE
 FAKIT(2*MUL)=FKSUM

A5
C OUTPUT A FILE FOR PLOTTING OF THE SMOOTHED RECONSTRUCTED
C DATA
0164 CALL PLTFLIL(FAKIT, 0, N, 1)
0165 WRITE(6, 120)
0166 11 FORMAT( 'COMB(A), EARLY(B), LATE(C), SELECT FILTER: ' )
0167 20 FORMAT( 'FILE ' )
0168 30 FORMAT( 'Q, 30A1')
0169 34 FORMAT( 'A1')
0170 106 FORMAT( 'INPUT NO SAMP PTS ')  
0171 109 FORMAT( 'I4')
0172 111 FORMAT( 'F10.5')
0173 112 FORMAT( 'E10.0')
0174 119 FORMAT( 'EXP. WEIGHTING FACT. ' )
0175 120 FORMAT( 'DONE')
0176 142 FORMAT( 'NO FILE PTS TO READ ') 
0177 210 FORMAT( '2X, 13, 2X, 2(E16.8, 2X)' )
0178 230 FORMAT( 'FOR ORIC DATA PLOT ONLY, TYPE 1 ' )
0179 999 END

PROGRAM SECTIONS

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIZE</th>
<th>ATTRIBUTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>CODE1</td>
<td>004444</td>
<td>1170</td>
</tr>
<tr>
<td>SPDATA</td>
<td>000934</td>
<td>14</td>
</tr>
<tr>
<td>IDATA</td>
<td>000462</td>
<td>153</td>
</tr>
<tr>
<td>VARS</td>
<td>011646</td>
<td>2515</td>
</tr>
<tr>
<td>TEMPS</td>
<td>000010</td>
<td>4</td>
</tr>
</tbody>
</table>

TOTAL SPACE ALLOCATED = 017040 3856

PPTEST, PPTEST/-SP=PPTEST
SUBROUTINE FFTDP(DATA, NN, ISGN)
C THIS SUBROUTINE TAKEN FROM R.H. BRENNER *THREE FORTRAN
C PROGRAMS THAT PERFORM THE COOLEY-TUKEY FOURIER TRANSFORM*
C MIT LINCOLN LAB TECH NOTE 1967-2
REAL DATA(2048), TEMPR, TEMPI, THETA, WR, WI
N=2*NN
J=1
DO 5 I=1, N, 2
IF(I-J) 1, 2, 2
1 TEMPR=DATA(J)
TEMPI=DATA(J+1)
DATA(J)=DATA(1)
DATA(J+1)=DATA(1+1)
DATA(1)=TEMPR
DATA(1+1)=TEMPI
N=N/2
IF(N-2) 5, 3, 3
J=J+N
N=N/2
IF(N-2) 5, 3, 3
5 J=J+N
MMAX=2
6 IF(MMAX-N) 7, 9, 9
7 ISTEP=2*MMAX
DO 8 N=1, MMAX, 2
WR=FLOAT(ISGN*(N-1))
WI=FLOAT(MMAX)
THETA=3.14159265358*WR/WI
WR=COS(THETA)
WI=SIN(THETA)
DO 8 I=M, N, ISTEP
J=1+MMAX
TEMPR=WR*DATA(J)-WI*DATA(J+1)
TEMPI=WR*DATA(J+1)+WI*DATA(J)
DATA(J)=DATA(1)-TEMPI
DATA(J+1)=DATA(1+1)-TEMPI
DATA(1)=DATA(1)+TEMPR
DATA(1+1)=DATA(1+1)+TEMPI
8 CONTINUE
MMAX=ISTEP
GO TO 6
9 CONTINUE
RETURN
END
SUBROUTINE LOWCD(N,HIDATA)
C THIS SUBROUTINE PROVIDES "EARLY-PASS" CATING. IT ZEROES
C ALL POINTS SYMMETRICALLY IN THE CEPSTRUM BETWEEN THE
C CUTOFF POINT AND THE CENTRAL POINT INCLUDING THE CUTOFF
C POINT.
REAL HIDATA(N)
WRITE(6, 1)
READ(5, 2) IRLY
DO 3 I=IRLY+2+1, N-(2*IRLY+1), 2
HIDATA(I)=OE0
3 CONTINUE
1 FORMAT('EARLY PASS CUTOFF ')
2 FORMAT(14)
RETURN
END

SUBROUTINE HICYD(N,HIDATA)
C THIS SUBROUTINE PROVIDES "LATE-PASS" CATING. IT ZEROES
C ALL POINTS SYMMETRICALLY BETWEEN THE END POINTS OF THE
C CEPSTRUM AND THE CUTOFF POINT. DOES NOT ZERO THE CUTOFF
C POINT.
REAL HIDATA(N)
WRITE(6, 1)
READ(5, 2) LATE
DO 3 I=1, LATE*2-3, 2
HIDATA(I)=OE0
3 CONTINUE
DO 4 I=N-(LATE*2-5), N, 2
HIDATA(I)=OE0
4 CONTINUE
1 FORMAT('LATE PASS CUTOFF ')
2 FORMAT(14)
RETURN
END
SUBROUTINE COMB(n, HIDATA)

C THIS SUBROUTINE GENERATES A GATING COMB FOR
C REMOVING THE SINGLE-POINT COMPONENTS OF THE IMPULSE
C TRAIN. "DELAY" SHOULD BE ENTERED AS THE NO. OF THE
C FIRST POINT IN THE IMPULSE TRAIN. "INTERVAL" SHOULD BE
C ENTERED AS THE NO. OF POINTS BETWEEN THE 1ST & 2ND &
C SUBSEQUENT IMPULSE POINTS. UP TO 10 SEQUENCES OR
C IMPULSE TRAIN VALUES MAY BE ENTERED. GATED POINTS ARE
C REPLACED BY THE MEAN VALUES OF THE NEIGHBORING
C POINTS TWO POINTS AWAY AND ON EITHER SIDE OF THE GATED
C POINT. STARTING-POINT SYMMETRY IS MAINTAINED AT THE ENDS
C OF THE CEPSTRUM.

DIMENSION IDLY(10), IDEL(10)
REAL HIDATA(1), FK0, FK1, FK3, FK4
BYTE IANS

K=1
WRITE(6,1)
READ(5,2) IDLY(K)
WRITE(6,3)
READ(5,2) IDEL(K)
WRITE(6,4)
READ(5,5) IANS
IF(IANS.EQ. 'N')GO TO 6
K=K+1
GO TO 7
6 DO 8 L=1, K
DO 9 J=2*IDLY(L)-1,N-(2*IDLY(L)-1),2*IDEL(L)
FK1=HIDATA(J-2)
FK3=HIDATA(J+2)
FK0=HIDATA(J-4)
FK4=HIDATA(J+4)
HIDATA(J)=(FK0-FK4)/2+FK4
CONTINUE
8 CONTINUE
9 FORMAT('#DELAY ')
2 FORMAT(14)
3 FORMAT('#INTERVAL ')
4 FORMAT('#MORE? (Y OR N) ')
5 FORMAT(CA1)
RETURN
END
L. B. POCHE, JR.

SUBROUTINE UF

This subroutine originally generated a 128-point time series function similar in appearance to Ulrych's seismic wavelet. It allowed keyboard inputs of echo amplitude, echo delay, and glitch amplitude. The expression used to generate the waveform was

\[ UF(t) = \frac{81 - (t - 55.8)^2}{270} \times \frac{(t - 66.5) (t - 41.5) e^{270/1155}}{1155} \]

The glitch was added at the 49th point and its echo at 49 + delay.

SUBROUTINE FILRED

This is a file-reading subroutine which is used to load input data from the selected file. It contains provision for keyboard inputs which specify the number of points to be discarded at the beginning of the data sequence and the sifting factor to be employed in sampling the data. The data are loaded into the array in alternate positions corresponding to the real parts of complex numbers.

SUBROUTINE PLTFIL

This subroutine writes data from an array of complex points as an output file to a storage device in a format compatible with the general-purpose library graphics program in use at this laboratory. It provides for selection from the keyboard of real, imaginary, or magnitude data to be written to the output file.

SUBROUTINE PLTFUL

This subroutine is similar to PLTFIL except that it is designed to output from a 512-point array of real data only.