

T-2082

VIBROKATOR

METHOD OF INTERPRETING SEISMOGRAMS

FROM VIBRATOR ENERGY SOURCES

by

George E. Handley

ProQuest Number: 11016487

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



ProQuest 11016487

Published by ProQuest LLC (2019). Copyright of the Dissertation is held by the Author.

All rights reserved.

This work is protected against unauthorized copying under Title 17, United States Code
Microform Edition © ProQuest LLC.

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

A thesis submitted to the Faculty and the Board of Trustees of the Colorado School of Mines in partial fulfillment of the requirements for the degree of Master of Science, Geophysics.

Signed: George E. Handley
George E. Handley

Golden, Colorado

Date: 13 June, 1978

Approved: J. E. White
J. E. White
Thesis Advisor

Golden, Colorado

Date: 13 June, 1978

Approved: G. V. Keller
George V. Keller
Head of Department

Golden, Colorado

Date: June 13, 1978

ABSTRACT

With the Vibrosais[®] technique, the output to a vibratory input is measured as a function of time. Crosscorrelating this output with the input yields the seismic reflection record. If this source-earth-geophone combination could be measured as a function of frequency, a reflection record in the time domain could be realized through the Fourier transform.

VIBROLOKATOR is the name for a proposal by I. S. Chichinin to obtain the real part of the Fourier transform by using a vibrator whose instantaneous frequency varies with time at a relatively slow rate. Since the source-earth-geophone system is causal, the seismic record can be computed from the real part alone. The Fourier transform of this array has a real part which is made up of Klauder wavelets and their Hilbert transforms. This method requires sweeps of very long duration in order to avoid reducing the amplitude of late reflections.

An extension of this method called VIBROLOKATOR II was proposed by Dr. J. E. White. This method uses two sweeps, an up sweep along the positive instantaneous angular frequency axis, and a down sweep along the negative instantaneous angular frequency axis. The Fourier transform of this seismic record has a real part and an imaginary

[®]
Trademark of Continental Oil Company

part. After countering the effects of filter amplitude and phase, we obtain the seismic record by summing the real and imaginary parts. This method provides more accurate and interpretable data than the VIBROLOKATOR method, with economic sweep times.

These methods require long sweep times as compared to conventional methods, but because the number of digits recorded is independent of sweep time, there is no increase of data handling. The long sweep times enhance the signal-to-noise ratio.

CONTENTS

Page	iii	ABSTRACT
Page	vi	List of Symbols
Page	vii	List of Figures
Page	xi	Aknowledgements
Page	1	INTRODUCTION
		VIBROLOKATOR
Page	3	Mathematical Basis
Page	8	Discussion of Filter
Page	13	Results of Analog Model
Page	19	Results of Computer Model
		VIBROLOKATOR II
Page	26	Mathematical Basis
Page	31	Results of Computer Model
Page	37	Discussion of Parameters
Page	52	CONCLUSIONS
Page	54	Appendix A: "VIBROLOKATOR II-- A Vibratory Seismic Method" J. E. White and George Handley
Page	68	Appendix B: Users Manual Introduction
Page	76	Program Explanations
Page	107	Appendix C: Programs
Page	139	Appendix D: References

List of Symbols

f_i = instantaneous frequency
 t = sweep time
 a = alpha = $\pi df_i/dt$
 q = instantaneous angular frequency = $2\pi f_i = 2at$
 w = customary angular frequency, ie. $F(w) \leftrightarrow f(t)$
 f = customary frequency = $w/2\pi$
 t_r = time of reflection r
 T = record time
 $h(t)$ = filter function
 $A \exp(-j\theta)$ = Fourier spectrum of filter
 A_r = amplitude effect of the filter upon a particular reflection
 θ_r = phase shift introduced by the filter upon a particular reflection
 θ = total phase shift = $aT^2 + \theta$
 θ_r = phase shift of a particular reflection = $at_r^2 + \theta_r$
 FC = filter constant (= .99 for all models)
 k = filter coefficient = $\ln(FC)/\text{sample rate}$
 h_r = reflection coefficient of a particular reflection
 $G(q)$ = Hanning truncator
 $d(T)$ = Dirac delta

List of Figures

- Page 4, Figure 1 Representation of Chirp and delayed chirp signals; f_i vs t
- Page 8, Figure 2 Representation of a Chichinen Wavelet
- Page 9, Figure 3 Z-plane diagram of filter response
- Page 11, Figure 4 Plot of amplitude effect of filter vs frequency
- Page 12, Figure 5 Plot of phase angle added by filter vs frequency
- Page 14, Figure 6 Representation of analog test geometries
- Page 16, Figure 7 Plot of recorded waveform for first analog test
- Page 17, Figure 8 Plot of Fourier transform of first analog test
- Page 18, Figure 9 Plot of Fourier transforms of all three analog tests
- Page 22, Figure 10 Representation of VIBROLOKATOR method of recording field data.
- Figure 10A Zero to 10Hz, 10 sec. sweep
- Figure 10B Product with one reflection at $t_r=1$
- Figure 10C Filtered product
- Page 23, Figure 11A Filtered product of 10-50Hz, 60 sec. sweep with 10 reflections at .1 sec. intervals from .1 sec. to 1.0 sec.
- Figure 11B Real part of Fourier transform of 11A

Page 24, Figure 12A Filtered product of 10-50Hz, 60 sec.
sweep with 10 reflections at .5 sec.
intervals from .5 sec. to 5.0 sec.

Figure 12B Real part of Fourier transform of 12A

Page 25, Figure 13A Filtered product of 10-50Hz, 600 sec.
sweep with 10 reflections at .5 sec.
intervals from .5 sec. to 5.0 sec.

Figure 13B Real part of Fourier transform of 13A

Page 27, Figure 14 Representation of VIBROLOKATOR II
method of recording field data.

Figure 14A -10 to 10Hz, 20 sec. sweep

Figure 14B Product with one reflection
at $t_r=1$ sec.

Figure 14C Filtered product

Page 30, Figure 14.5 Representation of cosine and sine
weighting functions.

Page 33, Figure 15A Filtered product of 50-10Hz, 60 sec.
sweep and 10-50Hz, 60 sec. sweep,
each with ten reflections at .1 sec.
intervals from .1 sec. to 1.0 sec.

Figure 15B Real part and imaginary part of
Fourier transform of 15A

Page 34, Figure 16A Real part of $15B \times \cos\theta/A$ and
imaginary part of $15B \times \sin\theta/A$

Figure 16B Sum of real and imaginary parts in 16A

Page 35, Figure 17A Filtered product of 50-10Hz, 60 sec.
sweep and 10-50Hz, 60 sec. sweep,

each with 10 reflections at .5 sec.
intervals from .5 sec. to 5.0 sec.

Figure 17B Real part and imaginary part of
Fourier transform of 17A

Page 36, Figure 18A Real part of $17B \times \cos\theta/A$ and
imaginary part of $17B \times \sin\theta/A$

Figure 18B Sum of real and imaginary parts in 18A

Page 39, Figure 19A Filtered product of 50-10Hz 30 sec.
sweep and 10-50Hz, 30 sec. sweep,
each with 10 reflections at .5 sec
intervals from .5 sec. to 5.0 sec.

Figure 19B Real part and imaginary part of
Fourier transform of 19A

Page 40, Figure 20A Real part of $19B \times \cos\theta/A$ and
imaginary part of $19B \times \sin\theta/A$

Figure 20B Sum of real and imaginary parts in 20A

Page 42, Figure 21A Non-filtered product of 50-10Hz, 60
sec. sweep and 10-50Hz, 60 sec. sweep,
each with 10 reflections at .5 sec.
intervals from .5 sec to 5.0 sec.

Figure 21B Real part and imaginary part of
Fourier transform of 21A

Page 43, Figure 22a Real part of $21B \times \cos(aT^2)$ and
imaginary part of $21B \times \sin(aT^2)$

Figure 22B Sum of real and imaginary parts in 22A

Page 44, Figure 23A Non-filtered product of 50-10Hz, 30
sec. sweep and 10-50Hz, 30 sec. sweep,

each with 10 reflections at .5 sec.
intervals from .5 sec. to 5.0 sec.

Figure 23B Real part and imaginary part of
Fourier transform of 23A

Page 45, Figure 24A Real part of 23B x $\cos(aT^2)$ and
imaginary part of 23B x $\sin(aT^2)$

Figure 24B Sum of real and imaginary parts in 24A

Page 46, Figure 25 Distribution of amplitude along
record time from 10-50Hz, 60 sec
sweep sampled at 40ms.

Page 47, Figure 26 Distribution of amplitude along
record time from 10-50Hz, 60 sec
sweep sampled at 8ms.

Acknowledgements

Much of the credit for this thesis should go to I. S. Chichinin, whom I have never met. It was his curiosity over the Vibroseis[®] technique that generated this new idea. Much of the credit should also go to Dr. J. E. White, who brought this idea back with him from Russia, improved upon it, and proposed it to me as a research project.

I would like to thank the members of my committee, Dr. J. E. White, Dr. Frank Hadsell, and Dr. Bill Schneider, for their help and assistance in analyzing and writing this thesis. While at the Colorado School of Mines I have been working under a research assistantship that was funded in my second year by the Integrated Geophysics Project. For this I am grateful.

I would like to thank the administration of the C. S. M. Computer Center for their cooperation, and I would like to express my great appreciation for all the patient help Dr. J. E. White has given me.

VIBROLOKATOR

Method of Interpreting Seismograms

from Vibrator Energy Sources

INTRODUCTION

Oft times the most uncontrollable or uncontrolled aspect of seismic exploration is the source energy generation. The two most common, and at present most effective methods, are dynamite and Vibroseis.[®]

With dynamite, an uncontrolled spectrum, assumed infinite, is input in an instant of time. A large amount of energy is input, and the processing steps are relatively easy. But the uncontrolled spectrum of energy yields an uncontrolled spectrum of noise, and safety and ecological factors make dynamite hard to work with.

Vibroseis[®] was introduced as a means of controlling the bandwidth and the duration of the energy, and hence the noise. This method is also safe and ecologically acceptable. But this method requires much technology and computer hardware to interpret. It also inputs relatively little energy into the ground. As a result, many sweeps have to be summed to increase the signal-to-noise ratio, and

[®] Trademark of Continental Oil Company

complete processing is usually done at a center.

The "VIBROLOKATOR Method of Interpreting Seismograms from Vibrator Energy Sources" presents a means by which greater amounts of energy from one sweep can be input into the ground and the processing made much easier. This method may also offer much better control of the spectrum by virtue of the fact that the sweeps are much slower.

VIBROLOKATOR

Mathematical Basis

In this paper reference will be made to instantaneous angular frequency (q) and customary angular frequency (w). Data is recorded along an instantaneous angular frequency axis and Fourier transformed to a distribution defined over record time (T).

The data that is recorded along the instantaneous frequency axis is made up of customary angular frequencies. These customary angular frequencies are also referred to as sum and difference frequencies and are affected by the Fourier spectrum of the filter used.

If two cosine waves are multiplied

$$\cos(A) \times \cos(B) = 1/2 [\cos(A+B) + \cos(A-B)]$$

the result is the cosine of the sum plus the cosine of the difference. Chichinin had the idea that if $\cos(A)$ was a real causal sweep signal and $\cos(B)$ a delayed, or reflected, signal, then the product record of these two would be a frequency domain representation of the earth response. This record would be made up of constant difference frequencies, characteristic of the linear frequency delay between the source and reflectors, and varying sum frequencies. After

folding, the record could be Fourier transformed as a function of instantaneous frequency to a distribution of time, with strong spikes from the supported difference frequencies and negligible contributions from the varying sum frequencies.

Consider $\cos(A)$ to be a pilot sweep, $\cos(at^2)$, of amplitude 1 ($a = dfi/dt = \alpha$). As in the causal chirp radar signal (Klauder, 1960) this sweep has instantaneous frequency modulated in a linear manner with time t . And consider $\cos(B)$ to be a delayed (reflected) signal, $hr \cos(a(t-tr)^2)$ of amplitude hr .

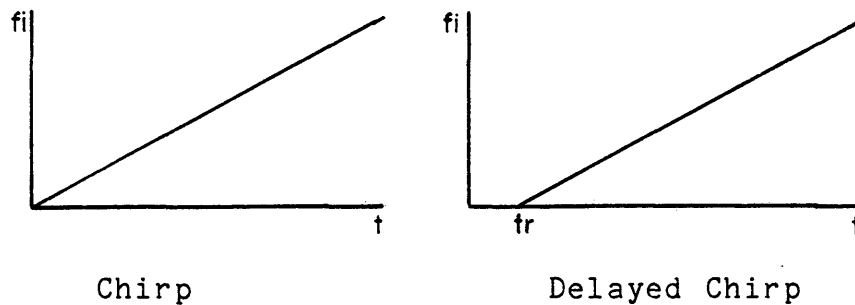


Figure 1

Representation of Chirp and Delayed Chirp Signals

The multiplication yields:

$$f(t) = \cos(at^2) \times hr \cos(a(t-tr)^2) =$$

$$(hr/2)[\cos(2at^2 - 2attr + atr^2) + \cos(2attr - atr^2)]$$

Varying Sum Frequency

Constant Difference Frequency

Multiplying the pilot signal with the delayed signal will, for $t \gg t_r$, give a record with a sum frequency component and a difference frequency component. If we choose to truncate this signal from 10 to 50Hz, for example, the instantaneous frequency of the sum component for each reflection will begin as low as 20Hz and increase to as high as 100Hz. The difference frequency component, however, will have a constant instantaneous frequency, and hence a customary frequency. This frequency will be proportional to the reflection time, or more specifically, equal to $2\alpha t_r \sin \theta$.

For a six second reflection, the continuous difference frequency components can vary from 8.00Hz to .40Hz with sweep times varying from 30 sec. to 600 sec. respectively (α varying from 30/40 to 600/40). A typical sample rate for the field is 4ms. For a 30 sec. sweep this represents 7,500 data points, and for a 600 sec. sweep this represents 150,000 data points. Either is a lot of data. If we stored only every fifth data point for the 30 sec. sweep, or every 100th data point for the 600 sec. sweep, this would give 1500 data points to transform and a 20ms or 400ms sample rate, respectively. This is adequate for sampling the difference frequencies from a 6 sec. reflection. The sum frequencies, which will add little more than noise, can be significantly diminished with a simple, in line, recursive filter.

Disregarding the sum frequency component we have

$$f(t) = (hr/2) \cos(2at - atr^2) * h(t).$$

It is a simple process to express the filter in terms of amplitude and phase lag. Using $wr = 2atr$

$$f(t) = (hr/2) \cos(wrt - atr^2) * h(t)$$

$$= (hr/4) [\exp(jwrt) \exp(-jatr^2) + \exp(-jwrt) \exp(jatr^2)] * h(t)$$

$$\leftrightarrow (\pi hr/2) A(wr) [\exp(-jatr^2) \exp(-j\theta(wr)) d(w+wr)$$

$$+ \exp(jatr^2) \exp(j\theta(wr)) d(w-wr)]$$

$$\leftrightarrow (hrAr/4) [\exp(-jatr^2 - j\theta r) \exp(jwrt)$$

$$+ \exp(jatr^2 + j\theta r) \exp(-jwrt)]$$

$$= (hrAr/2) \cos(wrt - atr^2 - \theta r)$$

Or, if $q = 2at$

$$F(q) = (hrAr/2) \cos(qtr - atr^2 - \theta r)$$

where $Ar \exp(-j\theta r)$ is the system function of the filter, expressed in amplitude and phase lag, acting upon the customary angular frequency (w) characteristic to the reflection r . Chichinin's proposed VIBROLOKATOR makes use

of an up-sweep in frequency only, which requires that t (and hence q) be positive. He recognized that the real part of the Fourier transform of this filtered product would approximate the seismic reflection record. One way to evaluate the real part of the transform is to consider the function to be made up of an even function and an odd function, and to transform the even part. An even function which equals the original function for $t > 0$ is:

$$F(q) = (hrAr/2) \cos(qtr \operatorname{sgnt} - atr^2 - \theta r)$$

Substituting $\theta r = atr^2 + \theta r$

$$\begin{aligned} F(q) &= (hrAr/2) \cos(qtr \operatorname{sgnt} - \theta r) \\ &= (hrAr/2) \cos(\operatorname{sgnt}(qtr - \theta r \operatorname{sgnt})) \\ &= (hrAr/2) \cos(qtr - \theta r \operatorname{sgnt}) \\ &= (hrAr/2) \cos(qtr - \theta r \operatorname{sgn}q) \\ &= (hrAr/4) [\exp(jqtr) \exp(-j\theta r \operatorname{sgn}q) \\ &\quad + \exp(-jqtr) \exp(j\theta r \operatorname{sgn}q)] \end{aligned}$$

Fourier transforming, (all real) with respect to q , and using T as the transform variable.

$$\begin{aligned}
 f(T) &= (hrAr/4)(1/2\pi) \int_{-\infty}^{\infty} [\exp(jqtr) \exp(-j\theta r \operatorname{sgn}q) \\
 &\quad + \exp(-jqtr) \exp(j\theta r \operatorname{sgn}q)] \exp(jqT) dq \\
 &= (hrAr/4) [d(T+tr) * (\cos(\theta r)d(T) - \sin(\theta r)(-1/\pi T)) \\
 &\quad + d(T-tr) * (\cos(\theta r)d(T) + \sin(\theta r)(-1/\pi T))]
 \end{aligned}$$

This is an even distribution representing the reflection record, weighted by $\cos(\theta r)$, plus it's Hilbert transform weighted by $\sin(\theta r)$ (White, 1965).

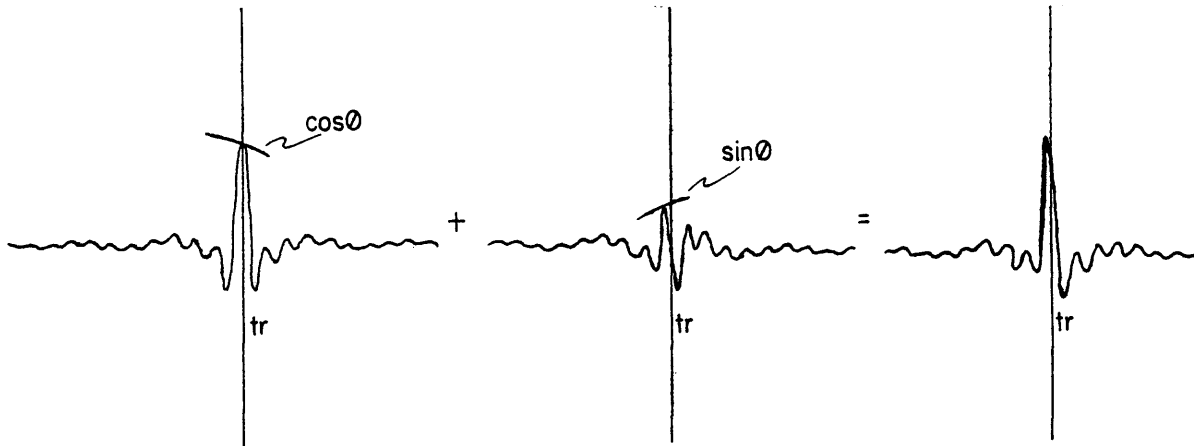


Figure 2

Representation of Chichinin Wavelet

The name given to this wavelet is the "Chichinin Wavelet". By definition, this is a distribution made up of the Klauder wavelets $\times \cos\theta$ plus the Hilbert transforms of the Klauder wavelets $\times \sin\theta$.

VIBROLOKATOR

Discussion of Filter

The filter used is a simple in line integrating filter whose recursion equation is

$$Y_n = X_n + FC(Y_{n-1})$$

where FC is the "filter constant" (Shanks, 1967). FC = .99 was used in all computer models.

In the Z-domain the transfer function of this filter is

$$F(z) = 1/(1-.99z)$$

This is a low pass filter with a pole at (1.0101+j0.0).

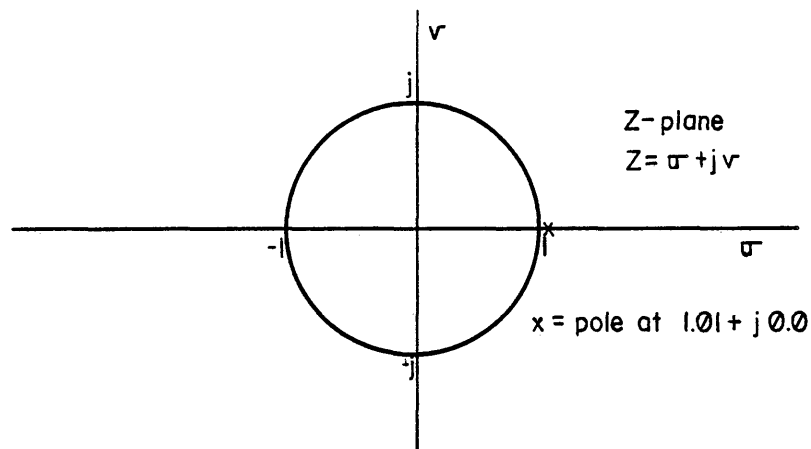


Figure 3

Z-plane diagram of filter response.

Converting the filter constant to an exponential and looking

at the filter function in the frequency domain (customary angular frequency) we have

$$.99 = \exp(-k\Delta t) \dots \text{if } t = 4\text{ms}, k = 2.5126$$

$$\begin{aligned} F(\omega) &= 1/(1 - \exp(-k\Delta t)\exp(-j\omega\Delta t)) \\ &= 1/(1 - \exp(-(k+j\omega)\Delta t)) \end{aligned}$$

Expressed in terms of amplitude and phase, this is

$$F(\omega) = A \exp(-j\theta)$$

where, if $|k+j\omega|\Delta t \ll 1$

$$A = (k^2 + \omega^2)^{-\frac{1}{2}}$$

$$\text{and } \theta = \tan^{-1}(\omega/k)$$

A plot (figure 4) of the normalized filter amplitude effect (kA) vs frequency reveals that only frequencies below .40Hz are above the 3db point. With this in mind we require a 600 sec. sweep to keep the amplitude of a 6 sec. reflection above the 3db point. This plot also shows that for a sweep beginning at 10Hz, the lowest sum frequency (20Hz) is down 34db. Figure 5 is a plot of the phase shift introduced by the filter vs frequency. Frequency in these two plots refers to customary frequency.

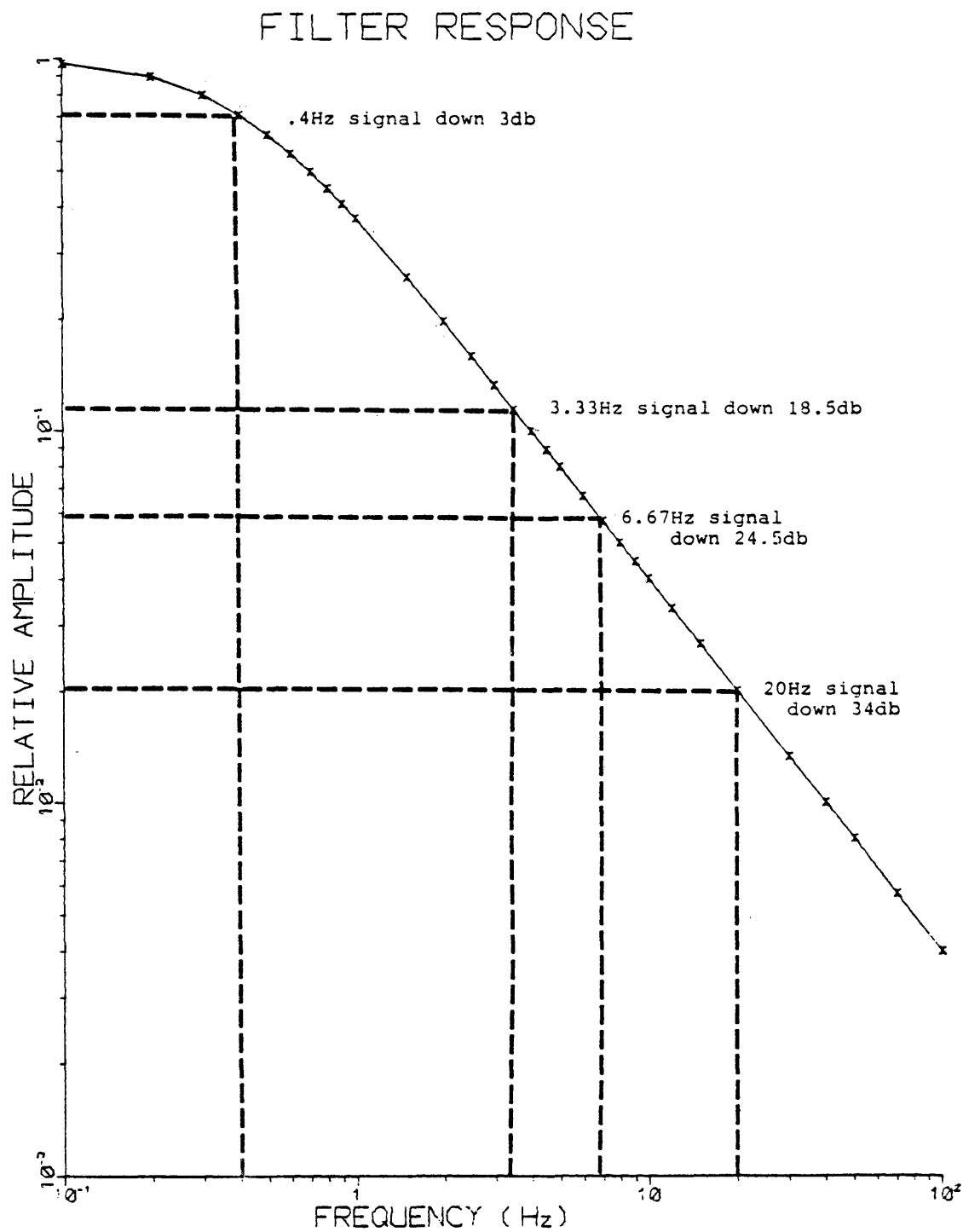


Figure 4 Amplitude effect of filter vs frequency

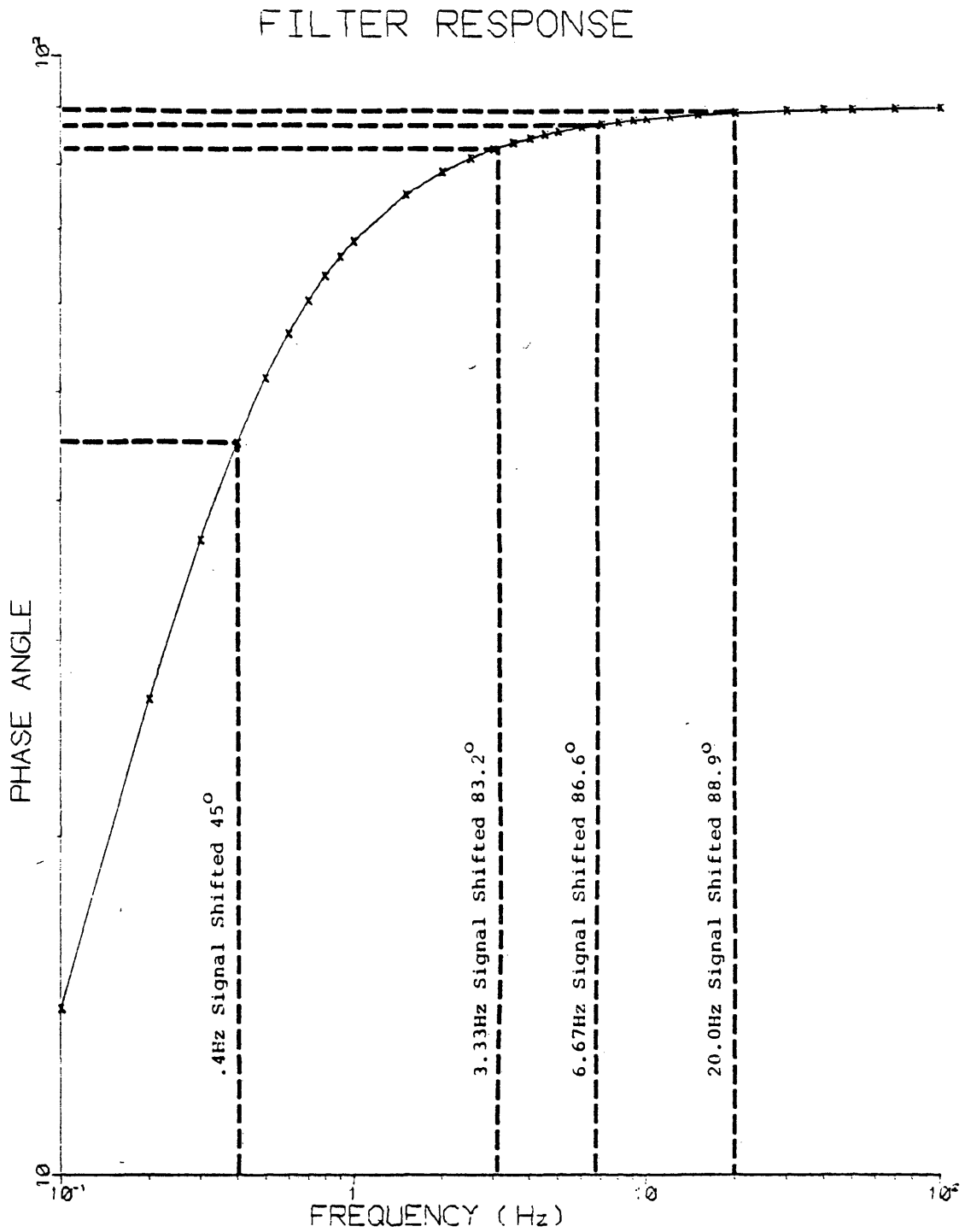


Figure 5 Phase angle added by filter vs frequency

VIBROLOKATOR

Results of Analog Model

An analog model was set up to test the VIBROLOKATOR method. A horn driver, a microphone and a reflecting surface were used in three different geometric arrangements. The audio sweep from a frequency generator ran from 700Hz to 2500Hz. The received signal was multiplied with the source, filtered through an integrating circuit, and sampled every 25Hz. in instantaneous frequency.

The geometry of the model was with the horn driver perpendicular to a reflecting surface four feet away. For the first reading the mike was centered between the source and the reflector. For the second reading the mike was moved six inches from center toward the source, and for the third reading the mike was moved six inches from center toward the reflector. It was anticipated that for each transformed output, two predominant spikes should appear on the record. One for the direct wave and one for the reflected wave. The resulting waveform was multiplied by a Hanning truncator.

$$\begin{aligned}
 G(f_i) &= \cos(\pi(1600-f_i)/900) & 700 < f_i < 2500 \\
 &= 0 & f_i < 700 \\
 &= 0 & f_i > 2500
 \end{aligned}$$

and inverse transformed using an FFT program. The results showed a ringing at about 1800Hz, so the data was multiplied by another window,

$$\begin{aligned}
 G'(f_i) &= 1.5 - \cos(2\pi(1800 - f_i)/900) & 700 < f_i < 2500 \\
 &= 0 & f_i < 700 \\
 &= 0 & f_i > 2500
 \end{aligned}$$

to suppress this frequency, and Fourier transformed.

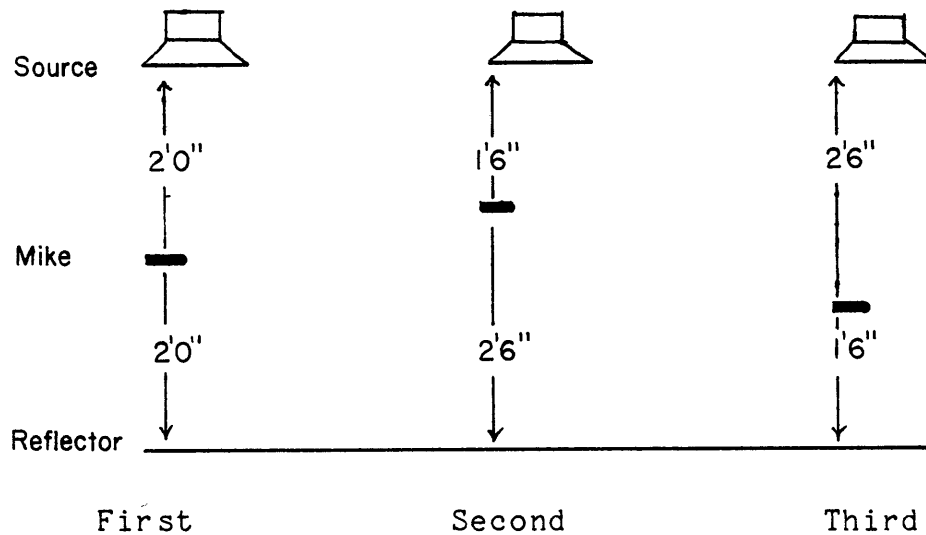


Figure 6

Representation of analog test geometries.

Figure 7 is a plot of the sampled waveform from the first reading. Figure 8 is the Fourier transform of the first reading. Figure 9 is the first 20ms of the Fourier transform of each reading. The top transform is from the third reading in which the microphone is furthest from the source and closest to the reflecting surface. The bottom

transform is from the second reading in which the microphone is closest to the source and furthest from the reflecting surface. The direct signal and the reflection are seen, at the proper times for travel through air.

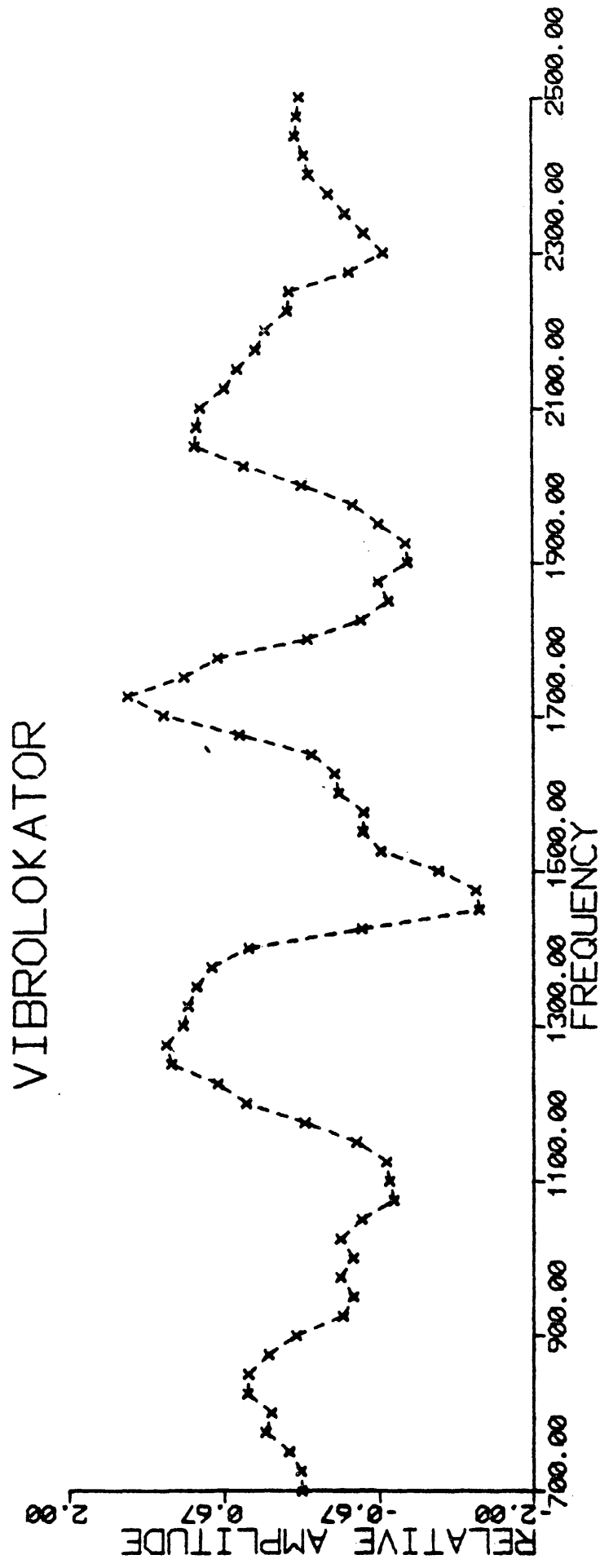


Figure 7 Plot of recorded waveform for first analog test.

VIBROLOKATOR

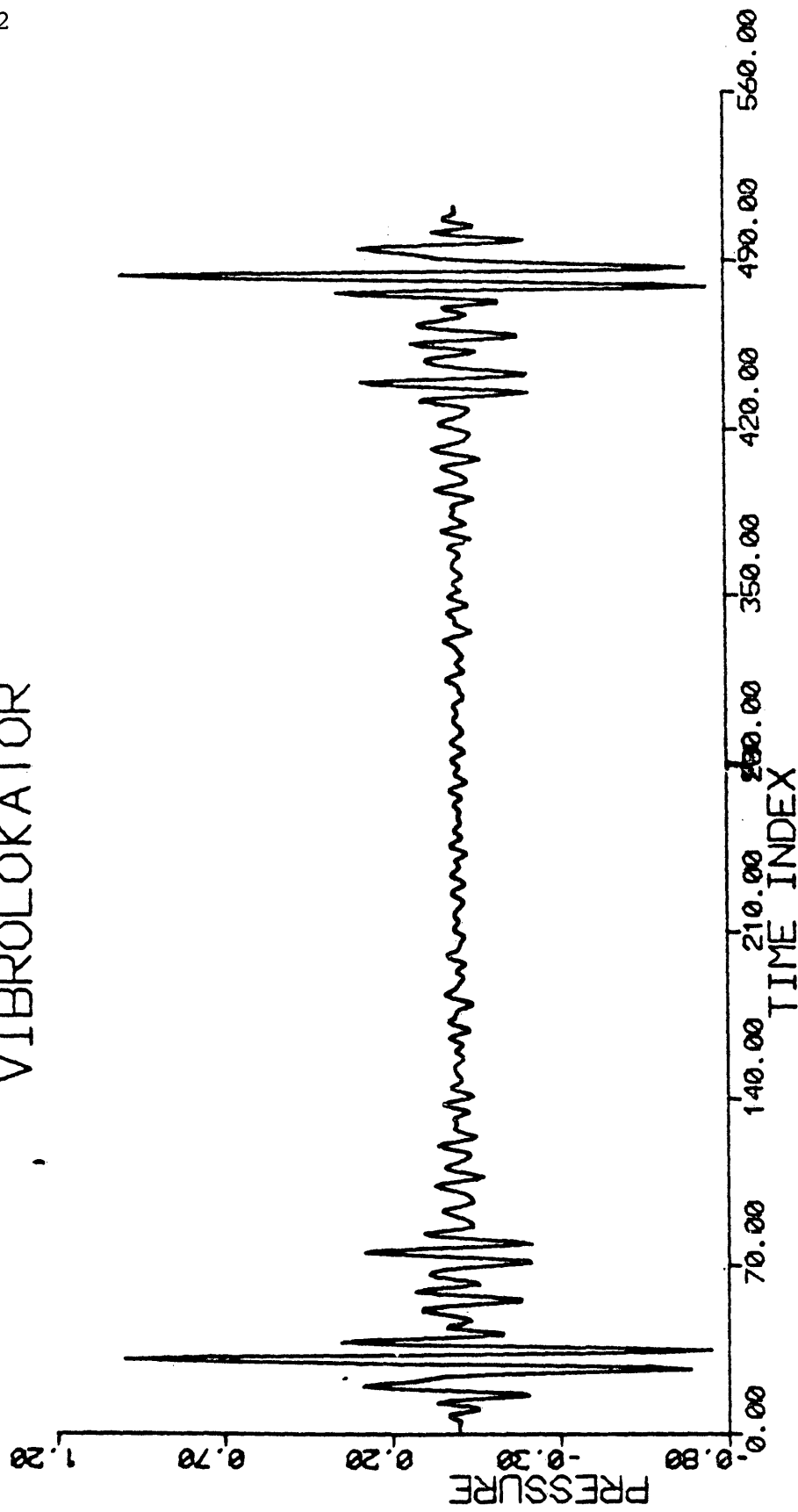


Figure 8 Plot of Fourier transform of first analog test.

VIBROLOKATOR

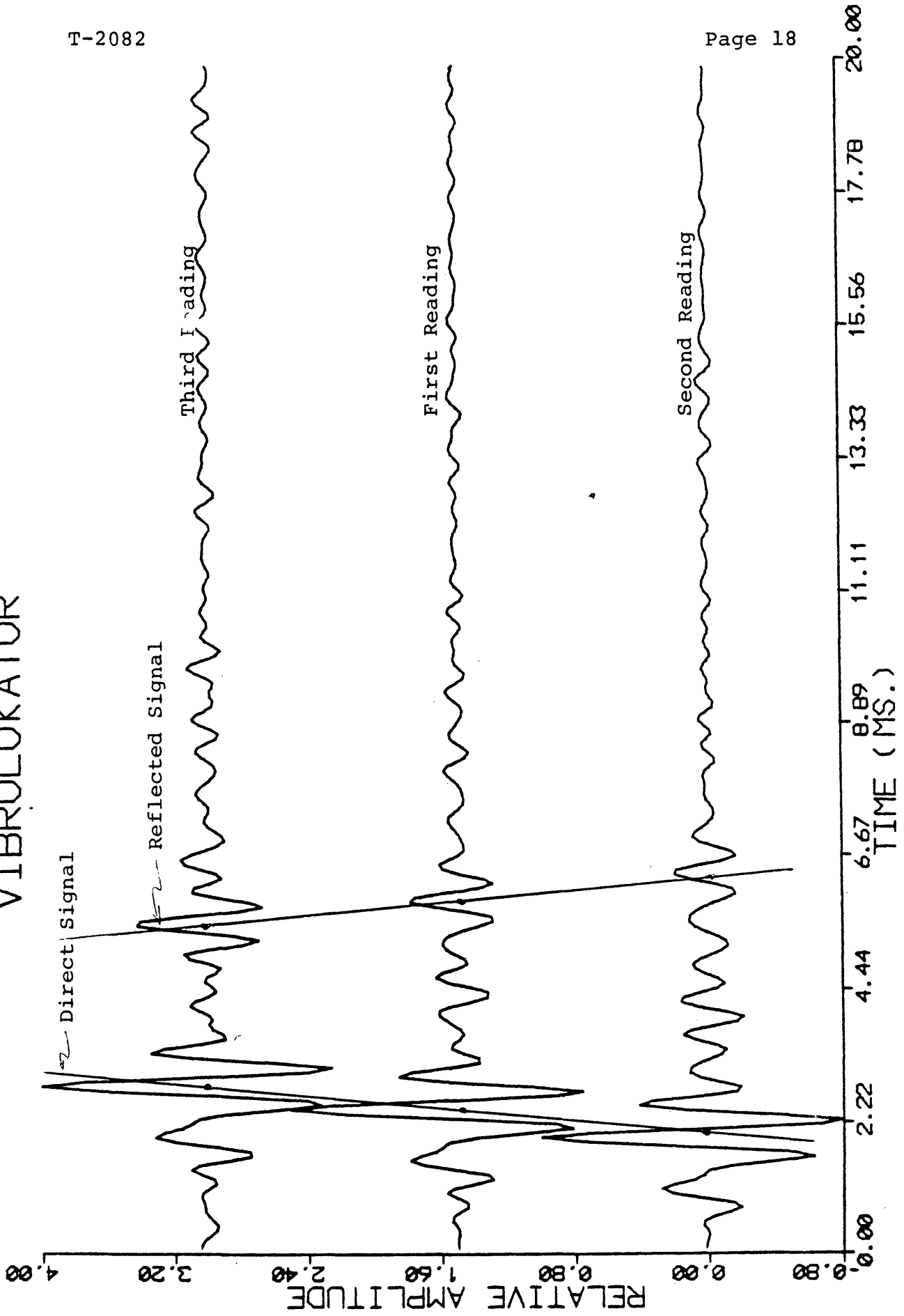


Figure 9 Plot of Fourier transforms of all three analog tests.

VIBROLOKATOR

Results of Computer Model

For a more exact analysis a digital computer model was generated with ten reflections. Parameters required by the program are:

Sweep Time
Initial Frequency
Final frequency
Sample Rate
Reflection Times
Filter Constant
Coupling Time
Sample Rate for Product

All constants and variables are determined by the program from the input parameters. The coupling time is the amount of time it takes for the amplitude of the pilot sweep to linearly increase from zero to one at the start of the sweep, and decrease from one to zero at the end. Each reflection is an exact replica of the pilot (amplitude of one), delayed by the reflection time t_r . For each time increment (typically 4ms) the value of each reflection is computed and the sum of these is multiplied with the pilot. The product is then input into the recursion equation and

the output sampled and stored.

The stored array is multiplied with a Hanning truncator $G(q)$. This causes each term of $f(T)$ on page 7 to be convolved with a zero phase wavelet $g(T)$. The array is then augmented with zeros, folded, and fast Fourier transformed (all real).

For a visual display, figure 10A is a plot of a pilot sweep from zero to 10Hz over 10 sec. sampled at 10ms. Figure 10B is the product of the pilot and a reflected signal at $t_r = 1$ sec. And figure 10C is the filtered product. Note the time lag introduced by the reflection ($-at_r^2$), and the time lag introduced by the filter ($-\theta r$). Note also that the sum and difference frequencies are not very well differentiated until about 4Hz.

Figure 11A is a 10 to 50Hz sweep over 60 sec. multiplied by the sum of sweeps from 10 reflections at .1 sec. intervals from .1 sec. to 1.0 sec. and then low pass filtered. The pilot and reflections were sampled at 4ms and the product at 40ms. Figure 11B is the Fourier transform of figure 11A.

The instantaneous phase function $\theta(T)$ is defined to equal θ_r at the reflection times (this will have more bearing later under the VIBROLOKATOR II discussion). Alpha for this case is $\pi/3$, so the cosine of θ should pass through a zero and the sine of θ a peak at

$$(\theta) = (aT^2 + \theta) = (\pi/2)$$

$$= (aT^2 + \tan^{-1}(2aT/k)).$$

This corresponds to $T = .61$ sec.

Note the near perfect asymmetry of the pulse at .6 sec. where the Hilbert transform of the wavelet is the only signal. After the cosine passes through zero the pulse becomes more symmetric and is inverted.

Figure 12A is the same sweep as figure 11A, but with the ten reflections at .5 sec. intervals from .5 sec. to 5.0 sec. and figure 12B is it's transform. Figure 13A is the same as 12A except with a 600 sec. sweep whose product was sampled at 400ms. (the same number of data points to be transformed). Alpha for this case is $\pi/15$ so θ is $\pi/2$ at 2.39 sec. Comparing this plot with figure 12 we can see that the effect of the filter on amplitude is much reduced for the longer sweep.

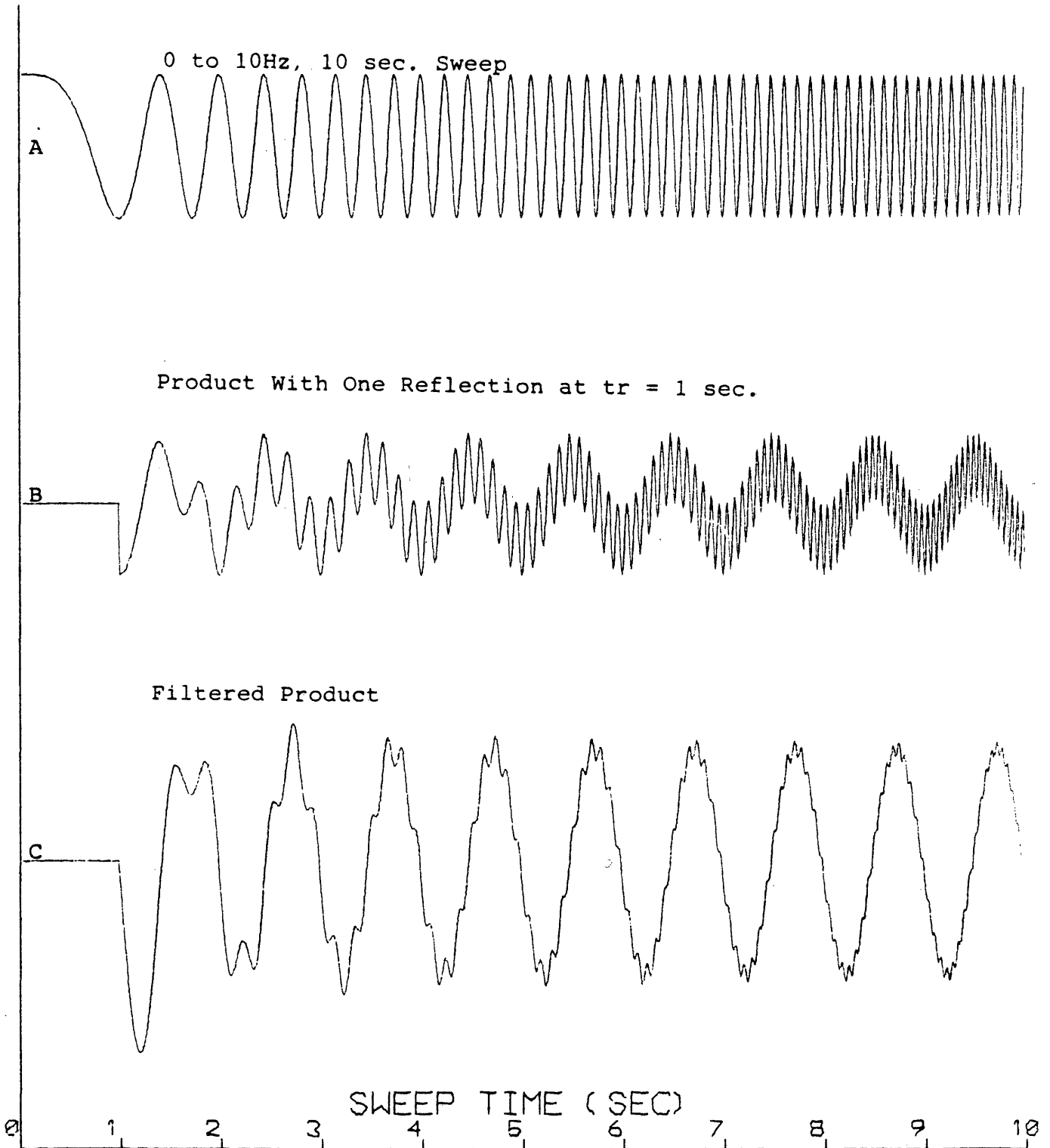
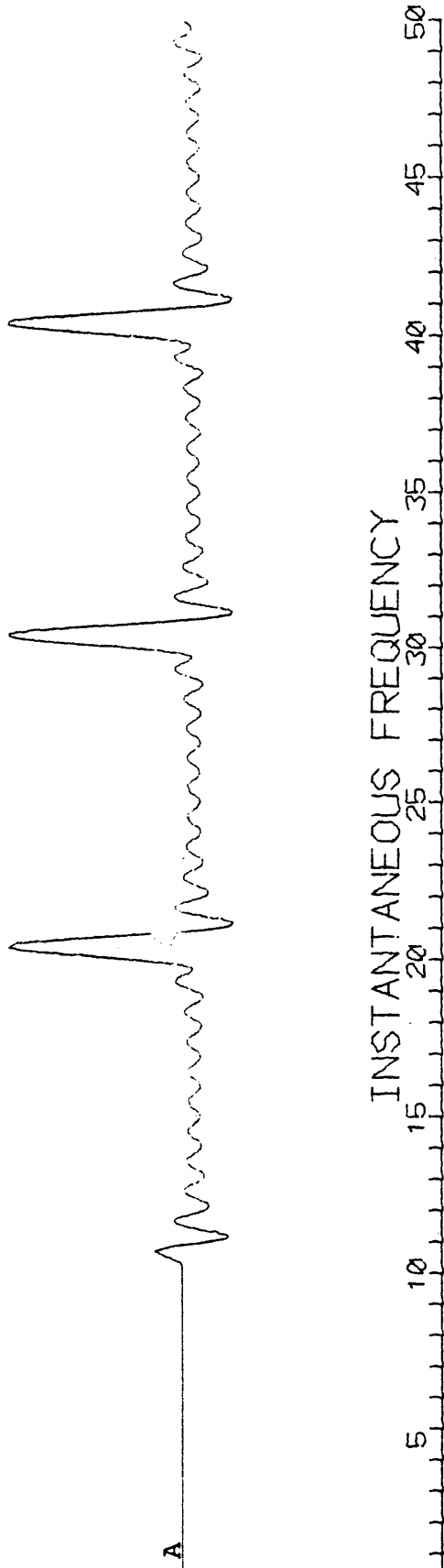


Figure 10A-C Representation of VIBROLOKATOR method of recording field data

Filtered Product of 10 to 50Hz, 60 sec. Sweep



Real Part of Fourier Transform

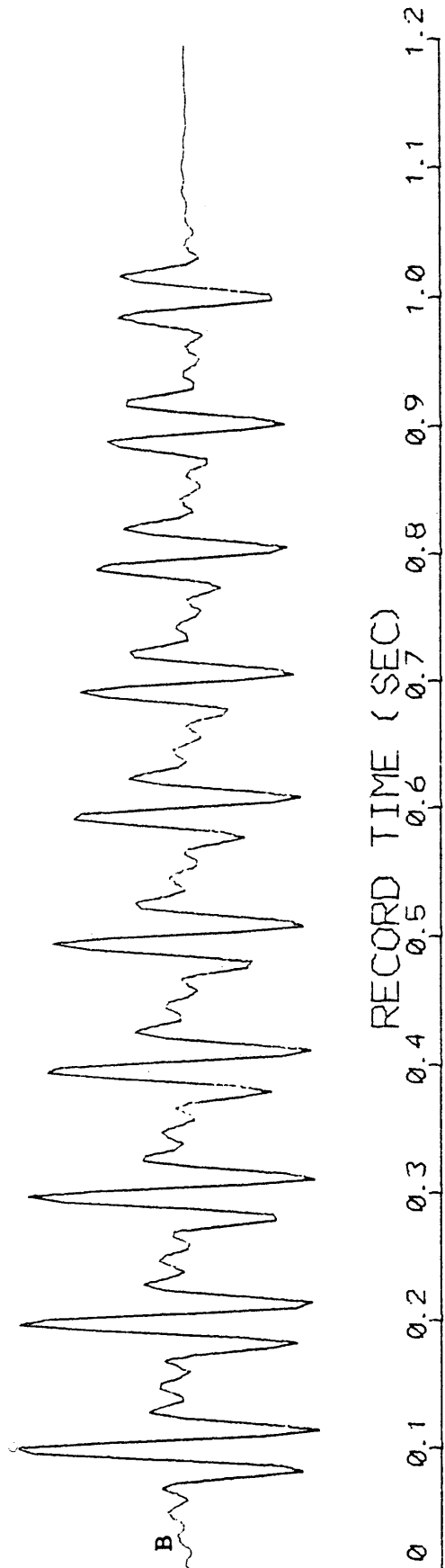
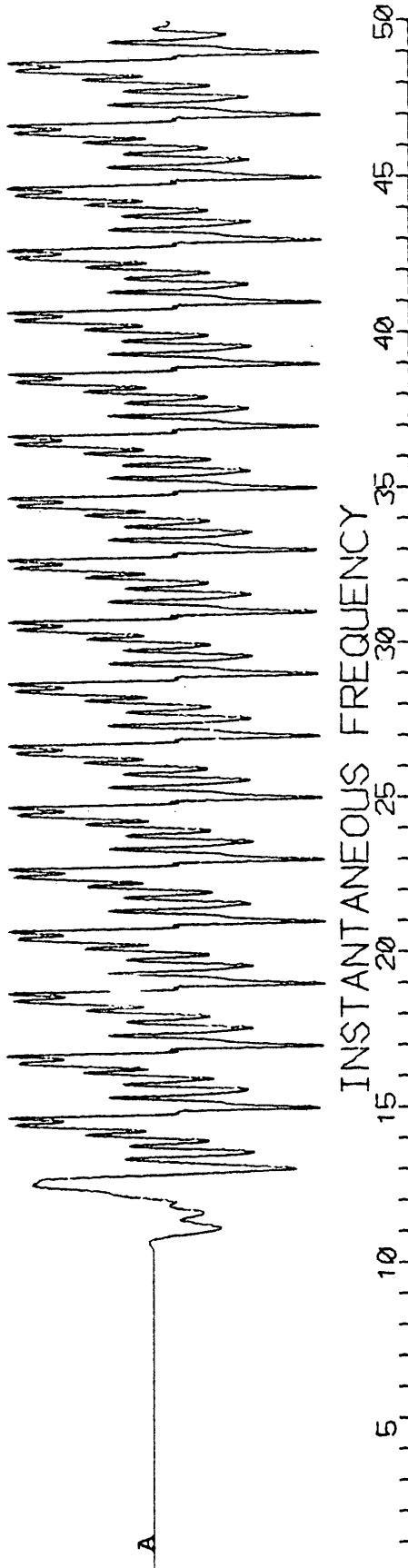


Figure 11A Filtered product of 10-50Hz, 60 sec sweep with 10 reflections at .1 sec intervals
11B Real part of Fourier transform of 11A

Filtered Product of 10 to 50Hz, 60 sec. Sweep



Real Part of Fourier Transform

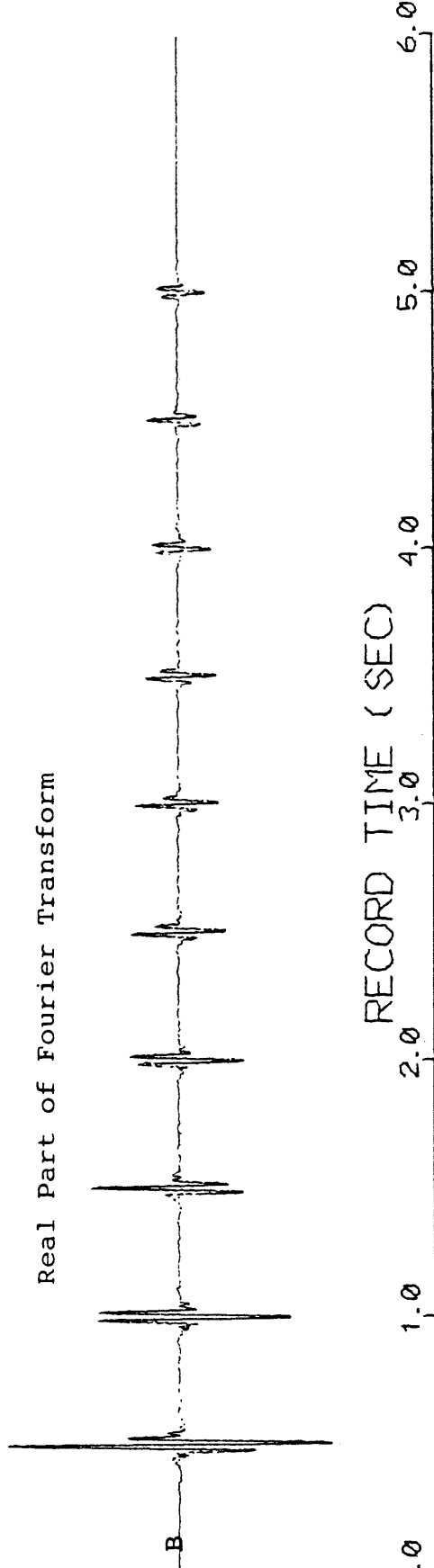
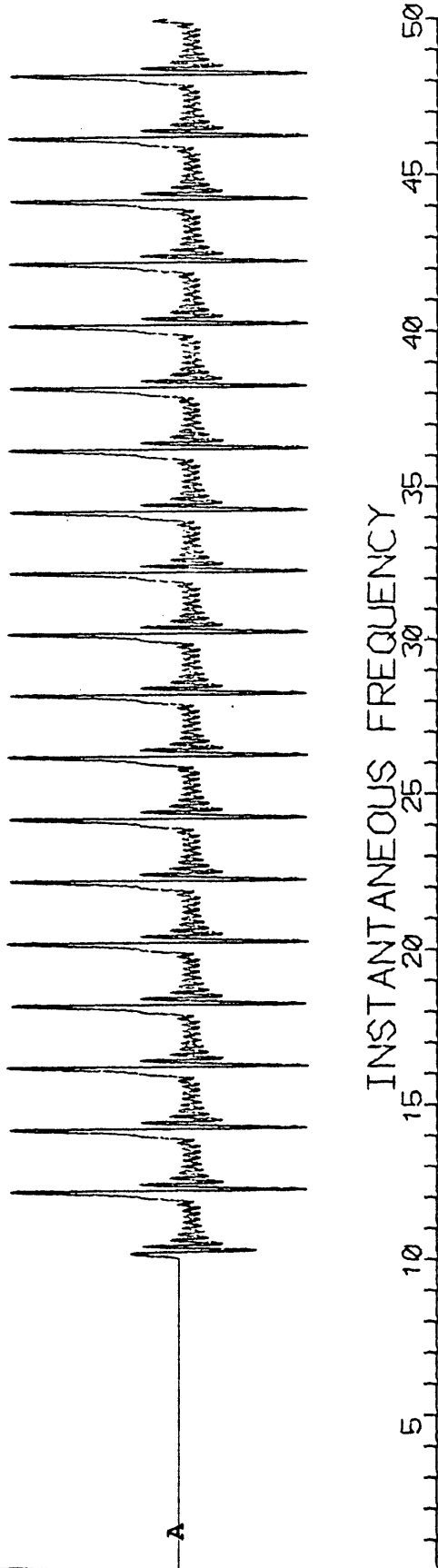


Figure 12A Filtered product of 10-50Hz sweep with 10 reflections at .5 sec intervals
12B Real part of Fourier transform of 12A

Filtered Product of 10 to 50Hz, 600 sec. Sweep



Real Part of Fourier Transform

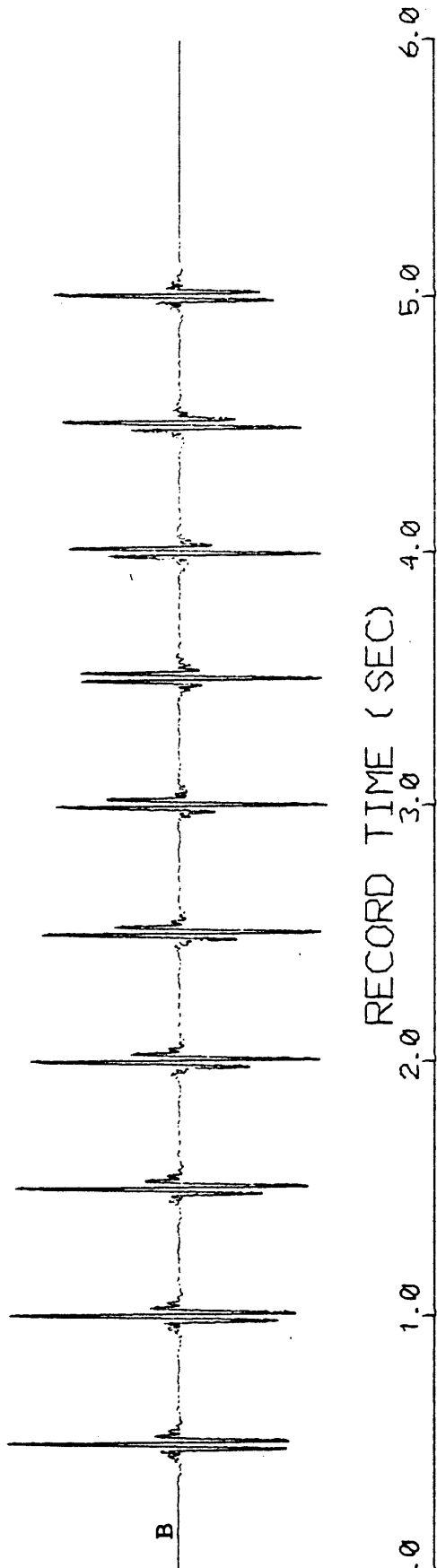


Figure 13A Filtered product of 10-50Hz, 600 sec sweep with 10 reflections at .5 sec intervals
13B Real part of Fourier transform of 13A

VIBROLOKATOR II

Mathematical Basis

J. E. White had the idea that a record with a down sweep and an up sweep would be a symmetric real record, defining the negative and positive instantaneous frequency axis. The product of the sweeps with a delayed signal would, as before, create an array with a constant difference frequency and a varying sum frequency, shifted by the amount (aT^2) . The filtered product would be the difference frequency only, with an additional phase shift (θ_r) .

This array, the filtered product of a down sweep and an up sweep is neither causal nor symmetric. The Fourier transform will now have a real part and an imaginary part, each made up of Klauder wavelets weighted by the cosine of the instantaneous phase and the sine of the instantaneous phase respectively.

Figure 14A represents a down sweep from 10 to zero Hz over 10 sec. plotted along the negative time axis, and an up sweep from zero to 10Hz over 10 sec. plotted along the positive time axis. Figure 14B represents this continuing down to up sweep multiplied by a reflected signal delayed 1 sec. And figure 14C is the filtered product.

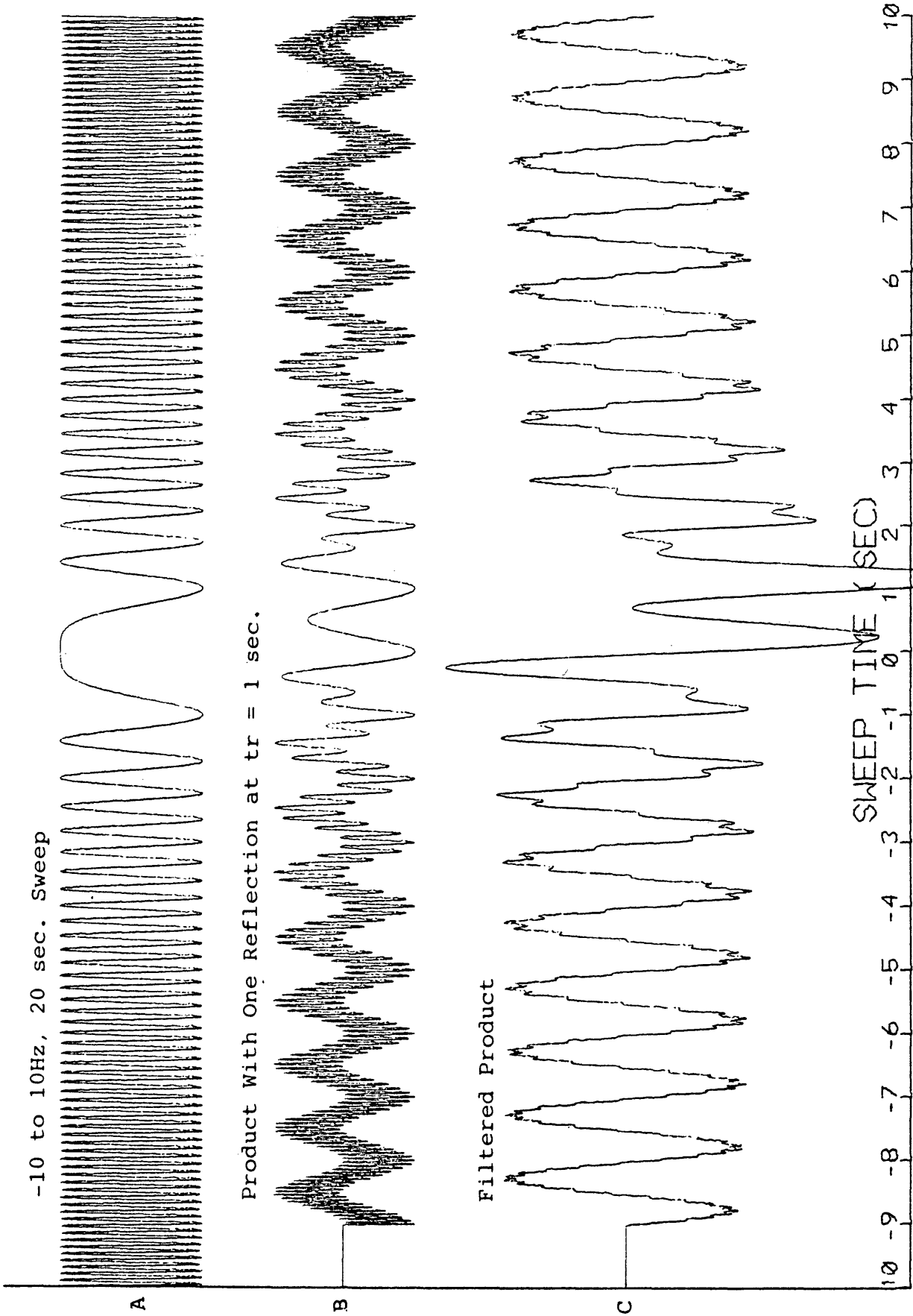


Figure 14A-C Representation of VIBROLOKATOR II method of recording field data

This combination of sweeps is, as described before, a linear sweep varying as $\cos(at^2)$ and its reflection, $\cos(a(t-tr)^2)$. But in this interpretation, t begins negative. The filtered product is approximately

$$f(t) = (hrAr/2)\cos(2attr - atr^2 - \theta r)$$

Expressing this in terms of q and applying the Hanning truncator,

$$\begin{aligned} F(q) &= G(q)(hrAr/2)\cos(qtr - \theta r). \\ &= G(q)(hrAr/4)[\exp(jqtr)\exp(-j\theta r) + \exp(-jqtr)\exp(j\theta r)] \end{aligned}$$

Fourier transforming:

$$\begin{aligned} f(T) &= (hrAr/4)(1/2\pi) \int_{-\infty}^{\infty} G(q)[\exp(jqtr)\exp(-j\theta r) + \\ &\quad \exp(-jqtr)\exp(j\theta r)]\exp(jqT) dq \\ &= g(T)*(hrAr/4)[d(T+tr)\exp(-j\theta r) + d(T-tr)\exp(j\theta r)] \\ &= g(T)*(hrAr/4)[d(T-tr) + d(T+tr)]\cos(\theta r) \\ &\quad + g(T)*(hrAr/4)[d(T-tr) - d(T+tr)]j\sin(\theta r) \end{aligned}$$

The transformed function now has a real part weighted by a cosine term plus an imaginary part weighted by a sine term. These real and imaginary arrays consist of the basic wavelets at times $T = \pm tr$. If we could square the cosine and sine terms, multiply by the inverse of the filter amplitude effect ($1/Ar$), and sum, we would arrive at the wavelets weighted only by their reflection coefficients.

θ_r , as mentioned earlier, is the phase angle due to the time of reflection (atr^2) plus the phase angle introduced by the filter acting upon that particular customary angular (difference) frequency (θ_r). Ar is the amplitude effect the filter has upon that frequency. These are both defined by time $T = tr$.

The instantaneous phase angle for any record time T was defined as

$$\theta = aT^2 + \tan^{-1}(2aT/k)$$

The amplitude effect for any record time T is defined as

$$A = [k^2 + (2aT)^2]^{-\frac{1}{2}}$$

Multiplying the real part by $(\cos\theta/A)$ yields a term proportional to

$$g(T-tr)(hr/4)\cos^2\theta_r$$

in the neighborhood of $T = tr$. Similarly, multiplying the the imaginary part by $(\sin\theta/A)$ yields a term proportional to

$$g(T-tr)(hr/4)\sin^2\theta r$$

This yields a close approximation to the wavelet if θ is varying relatively slowly (large alpha).

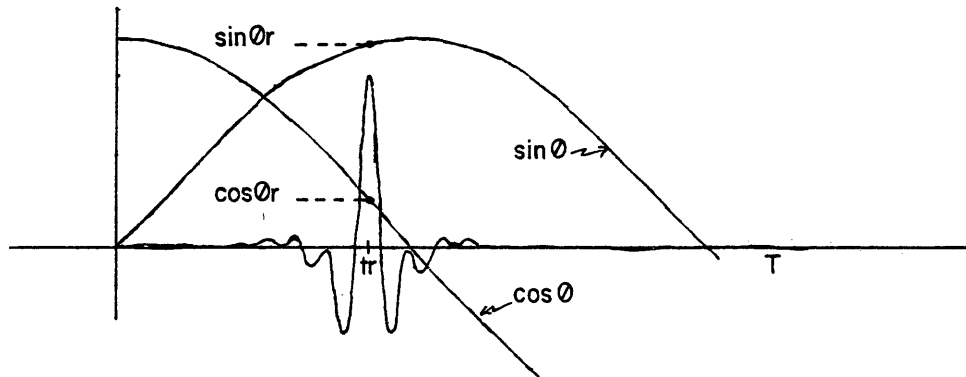


Figure 14.5

Representation of cosine and sine weighting functions.

As θ varies faster with shorter sweep times, the θ term begins to distort the wavelet to the point that the addition of the $\cos^2\theta$ and $\sin^2\theta$ terms do not yield a proportional wavelet. The sum therefore yields (for positive time only)

$$g(T-tr)hr/4.$$

This is the wavelet at time $T = tr$, weighted only by the reflection coefficient.

VIBROLOKATOR II

Results of Computer Model

This computer model is much like the model for the VIBROLOKATOR method. The up sweep is computed, summed, multiplied, filtered, and sampled as before. The down sweep is generated in the same manner, only with the instantaneous frequency of the pilot sweep decreasing from the "Final Frequency" to the "Initial Frequency". Both arrays are multiplied with a Hanning truncator and augmented with zeros. The up sweep is then loaded into the first half of the array to be transformed (all real), the down sweep is loaded into the folded half (all real), and the array is fast Fourier transformed. The real and imaginary parts of the transformed array are multiplied by their weighting factors and summed.

Figure 15A is a down sweep along the negative time axis and an up sweep along the positive time axis. Both are 60 sec. sweeps, 10 to 50Hz, with reflections at .1 sec. intervals from .1 sec. to 1.0 sec. The data was generated at a 4ms sample rate and the product sampled at 40ms. Figure 15B shows the real and imaginary parts of the transformed data. Note the cosine weighting factor for the real part and the sine weighting factor for the imaginary part.

Figure 16A shows the real part multiplied by $(\cos\theta/A)$ and the imaginary part multiplied by $(\sin\theta/A)$. Note that all wavelets have the same polarity and that the relative amplitudes are weighted only by $\cos^2\theta$ or $\sin^2\theta$. Figure 16B is the sum of the real and imaginary parts in 16A (multiplied by two).

This is the final result. After recording the filtered product of an up sweep and a down sweep, each as a function of instantaneous frequency, the reflection record in the time domain is realized through the Fourier transform. This reflection record has true relative amplitude (weighted only by the reflection coefficients) and waveform.

Figures 17 and 18 are the same sequence of figures as 15 and 16, but with the reflections at .5 sec. intervals. Note the filter amplitude effects in figure 17B that are corrected for in figure 18A.

The pilot sweeps and reflections within these models are in pure form without noise. The chatter on the output wavelet between pulses is due to aliased contributions from varying sum frequencies.

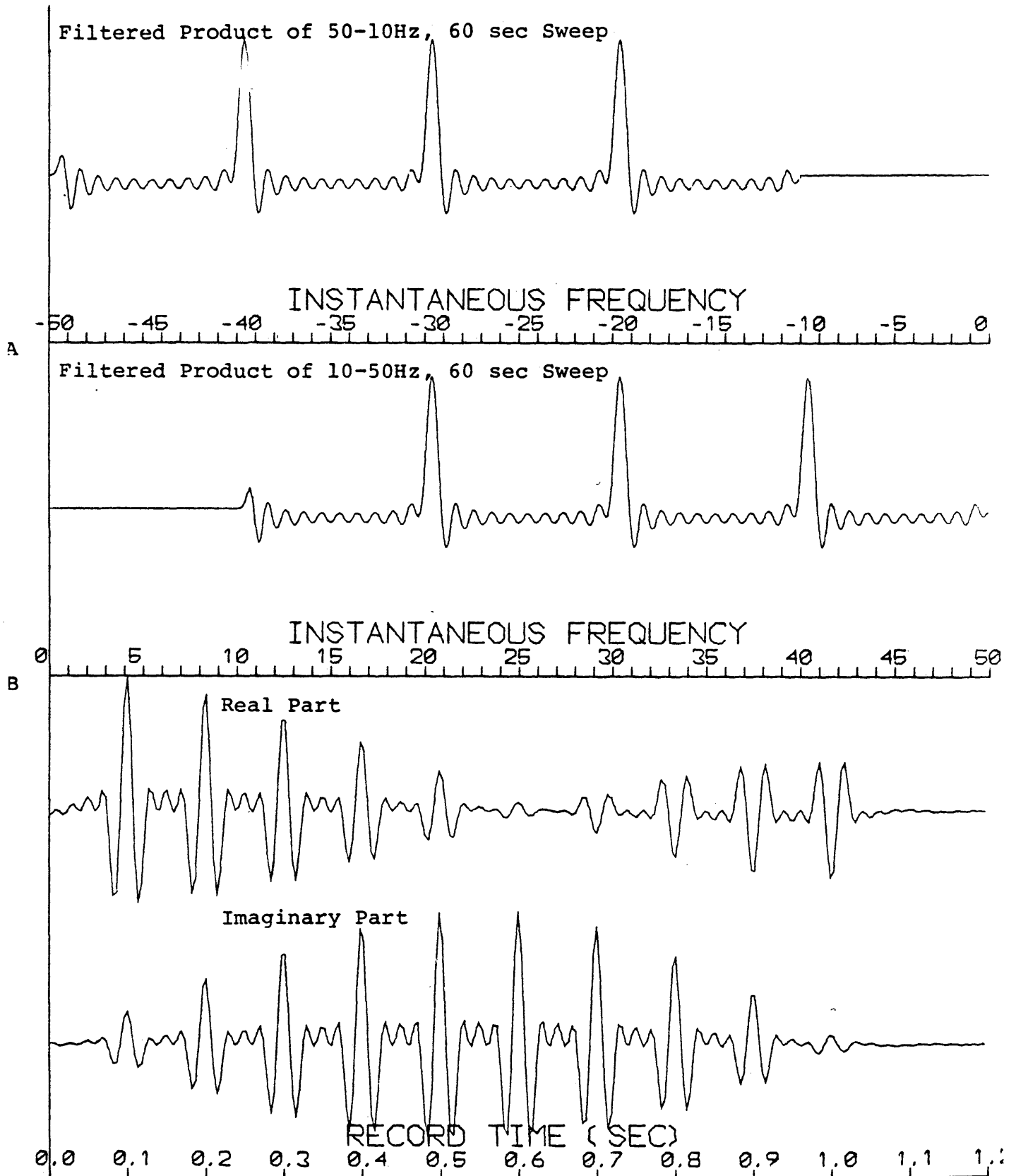


Figure 15A Filtered product of 50-10 and 10-50Hz, 60 sec sweeps with 10 reflections at .1 sec intervals from .1 sec to 1.0 sec
15B Real and imaginary parts of Fourier transform of 15A

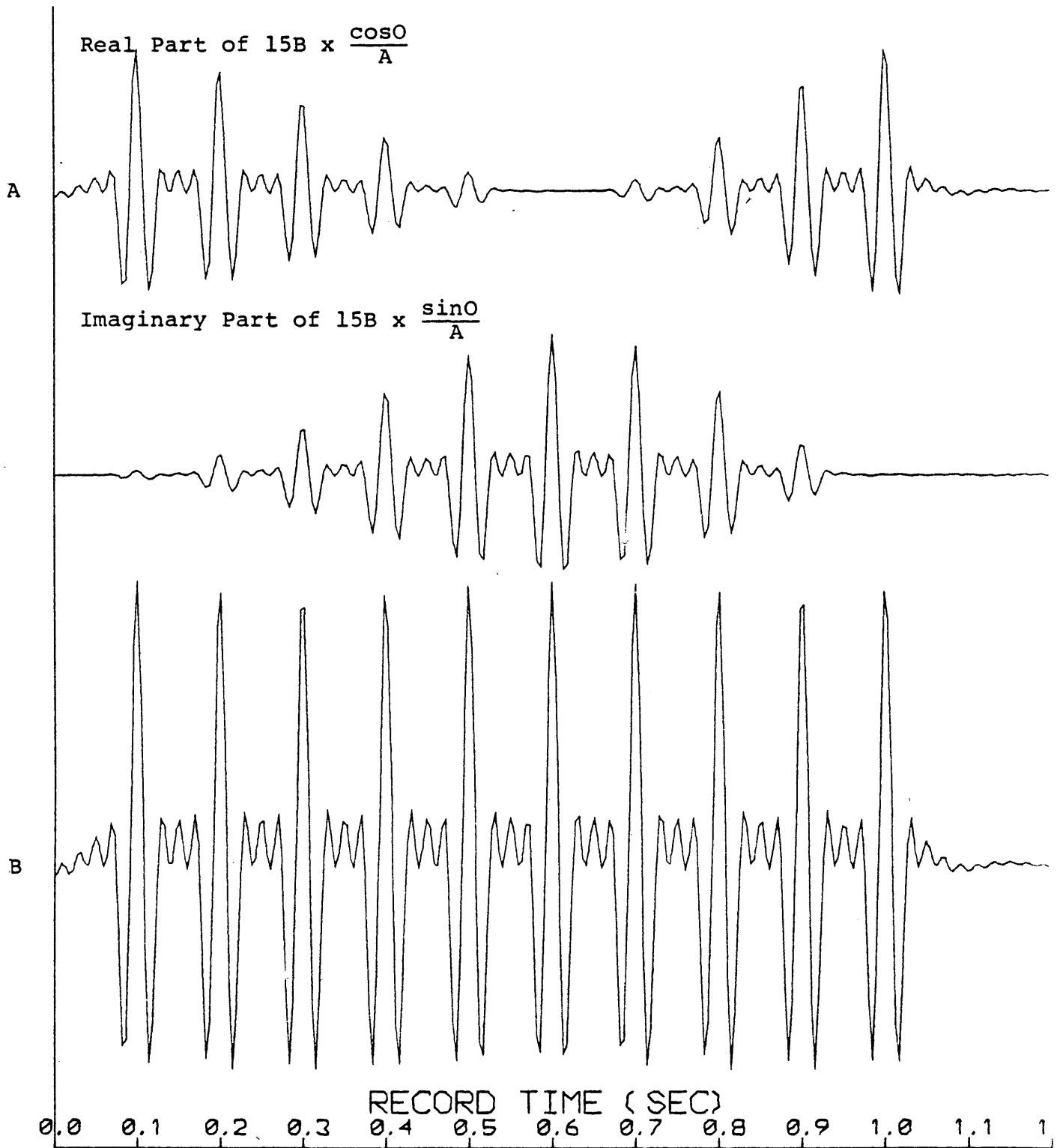


Figure 16A Real and imaginary parts of 15B multiplied by their respective weighting functions
16B Sum of real and imaginary parts in 16A (x2)

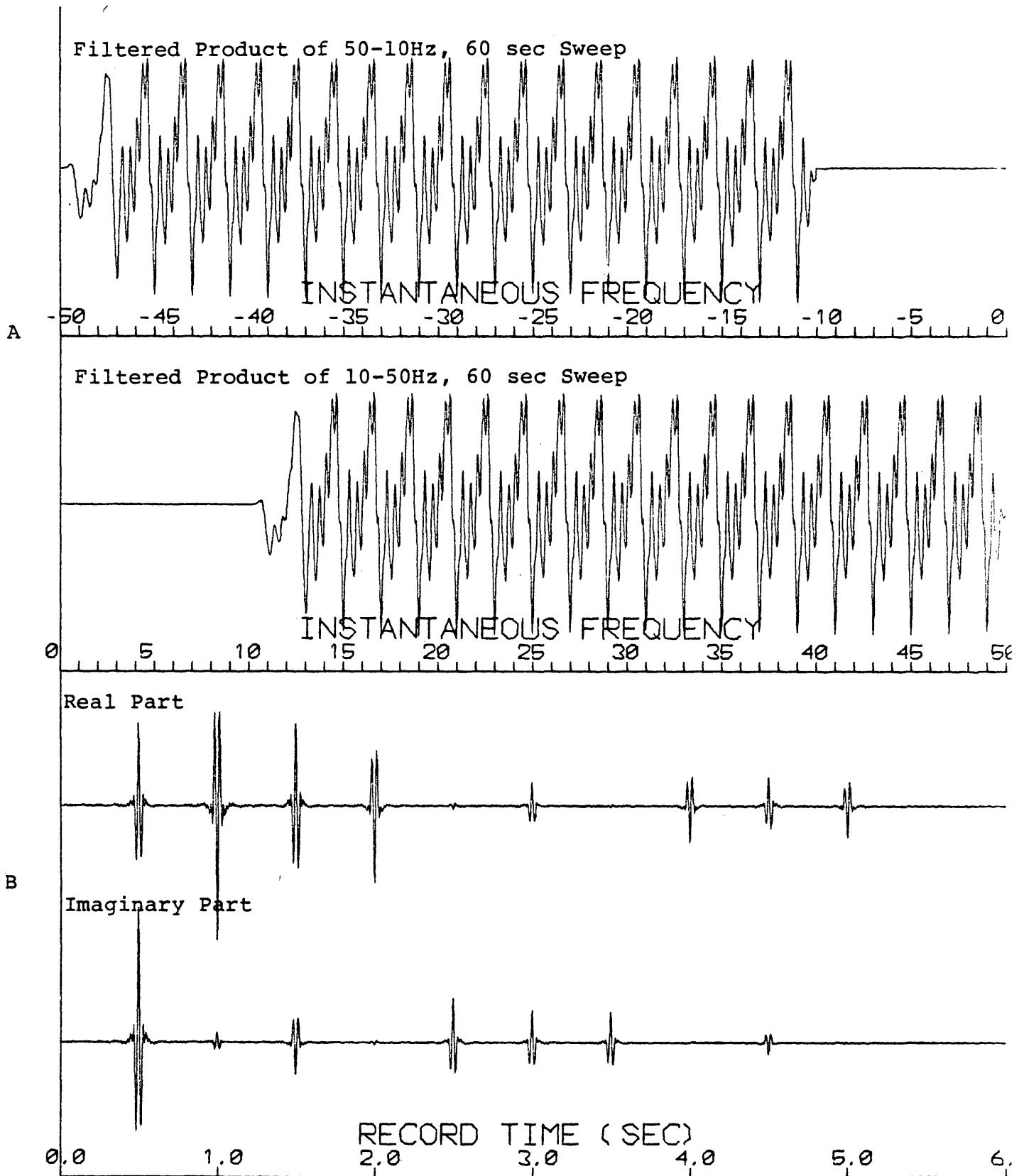


Figure 17A Filtered product of 50-10 and 10-50Hz, 60 sec sweeps with 10 reflections at .5 sec intervals from .5 sec to 5.0 sec
17B Real and imaginary parts of Fourier transform of 17A

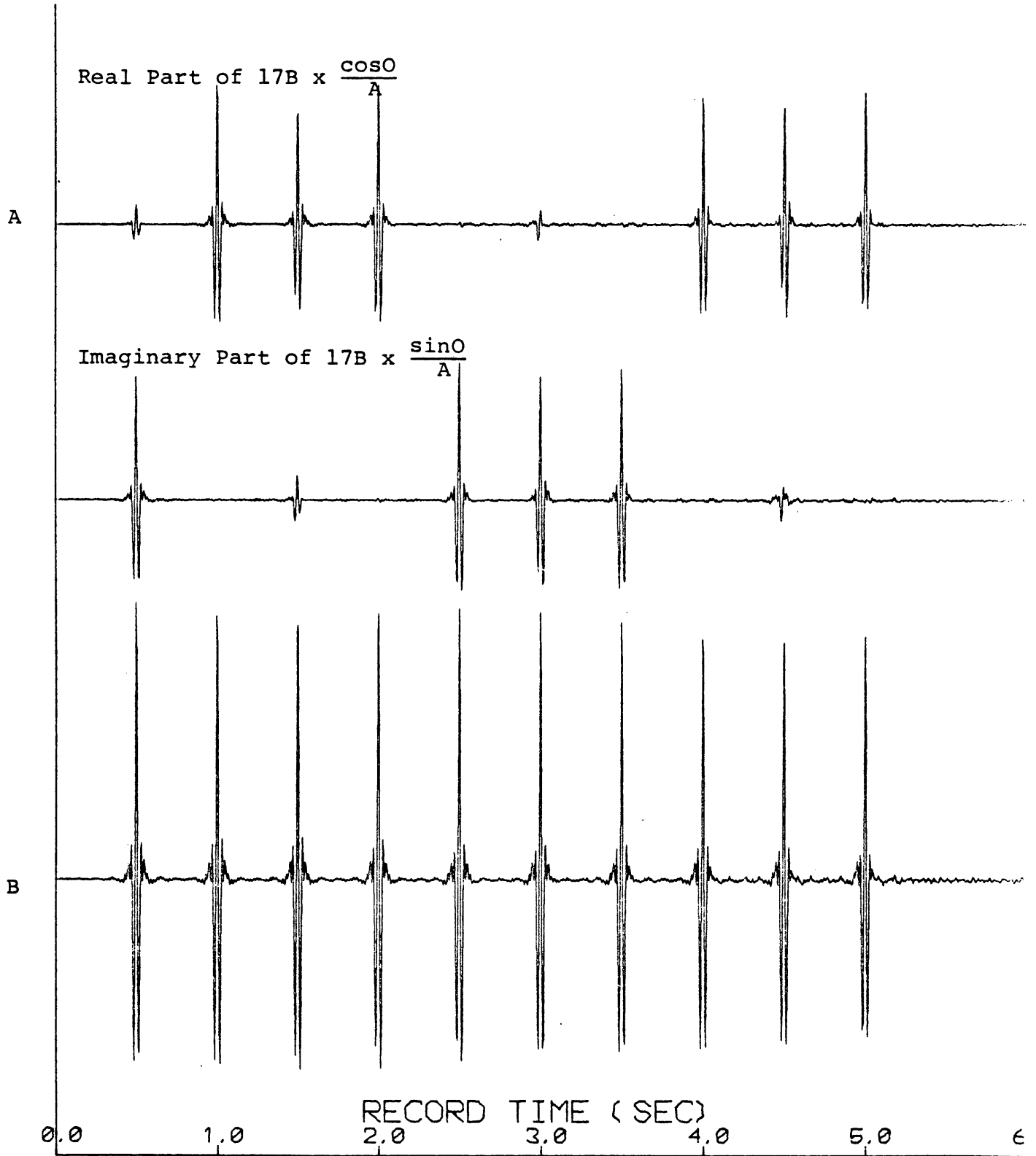


Figure 18A Real and imaginary parts of 17B multiplied by their respective weighting functions
18B Sum of real and imaginary parts in 18A (x 2)

VIBROLOKATOR II

Discussion of Parameters

The great advantage to the VIBROLOKATOR method is the signal to noise ratio that can be attained, and the ease of processing. With the VIBROLOKATOR II method of countering the filter amplitude and phase effects, more interpretable data can be gathered with economic sweep times.

From figure 4, the amplitude of a 5 sec. reflection for a 60 sec. 10 to 50Hz sweep (3.33Hz difference frequency) is down 18.5db. The amplitude of a 5 sec. reflection for a 30 sec. 10 to 50Hz sweep (6.67Hz difference frequency) is down 24.5db. And the amplitude of the lowest sum frequency (20Hz) is down 34.0db. With a ten decibel difference from a 5 sec. reflection difference frequency to the lowest sum frequency it seems feasible to run a 30 sec. sweep.

Figures 19 and 20 are the same sequence as figures 17 and 18, but with a 30 sec. sweep whose product was sampled at 20ms (the same number of data points to be transformed). Comparing figure 20 with figure 18 we see that there are a number of factors that determine the lower limit to sweep time. In rectifying the amplitude of a deep reflection for a 30 sec. sweep we are beginning to enhance the sum frequencies to the point of noise. This does not look that

bad on figure 20B, but remember the reflection coefficients for these models are all one. In actual practice they will be less than one, and substantially less for a five sec. reflection.

Also, with a 30 sec. sweep the cosine and sine terms oscillate much faster than with a 60 sec. sweep. As a result the weighting factor within one wavelet may change significantly, as evidenced in the wavelets of figure 20A.

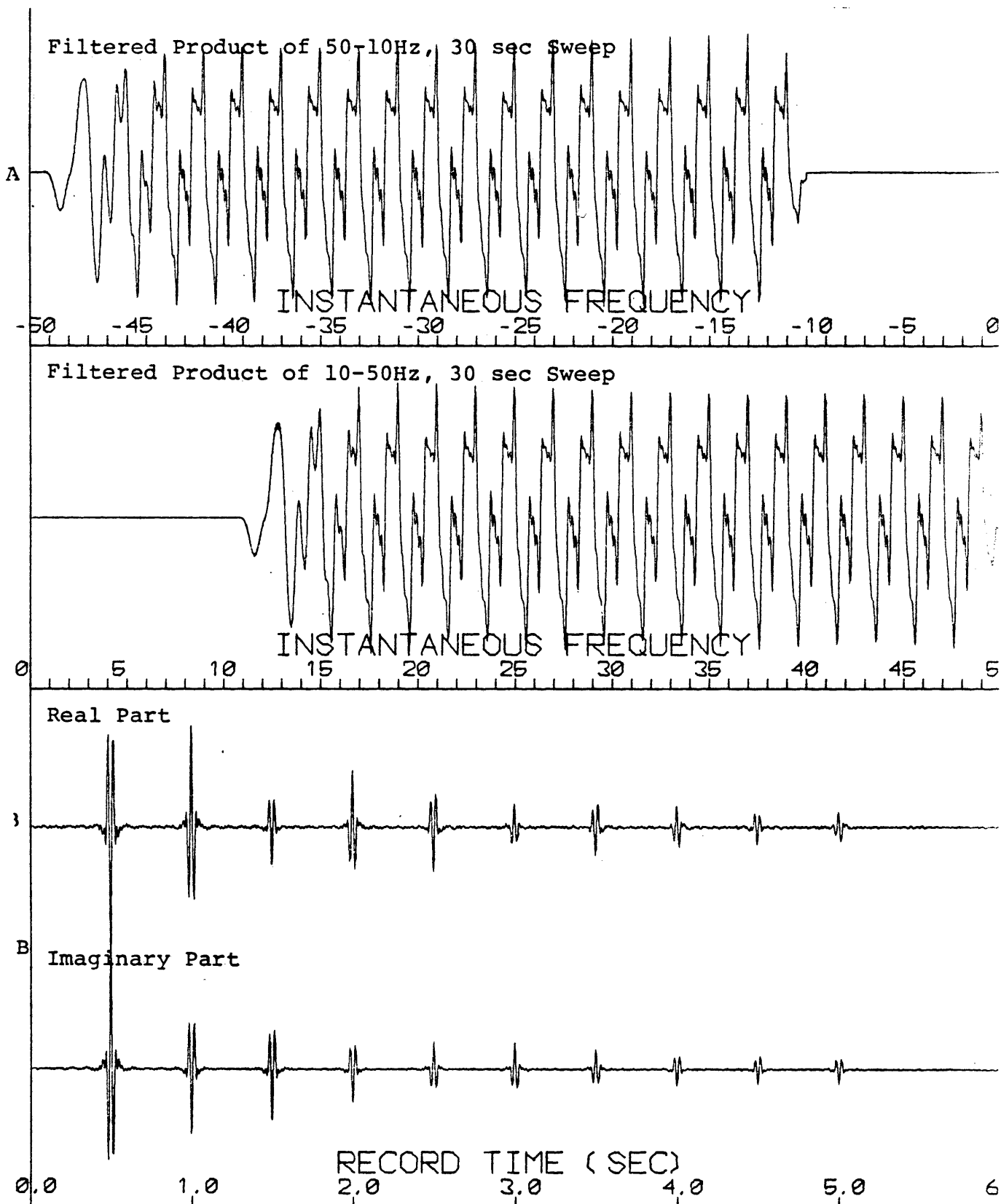


Figure 19A Filtered product of 50-10 and 10-50Hz, 30 sec sweeps with 10 reflections at .5 sec intervals from .5 sec to 5.0 sec
19B Real and imaginary parts of Fourier transform of 19A

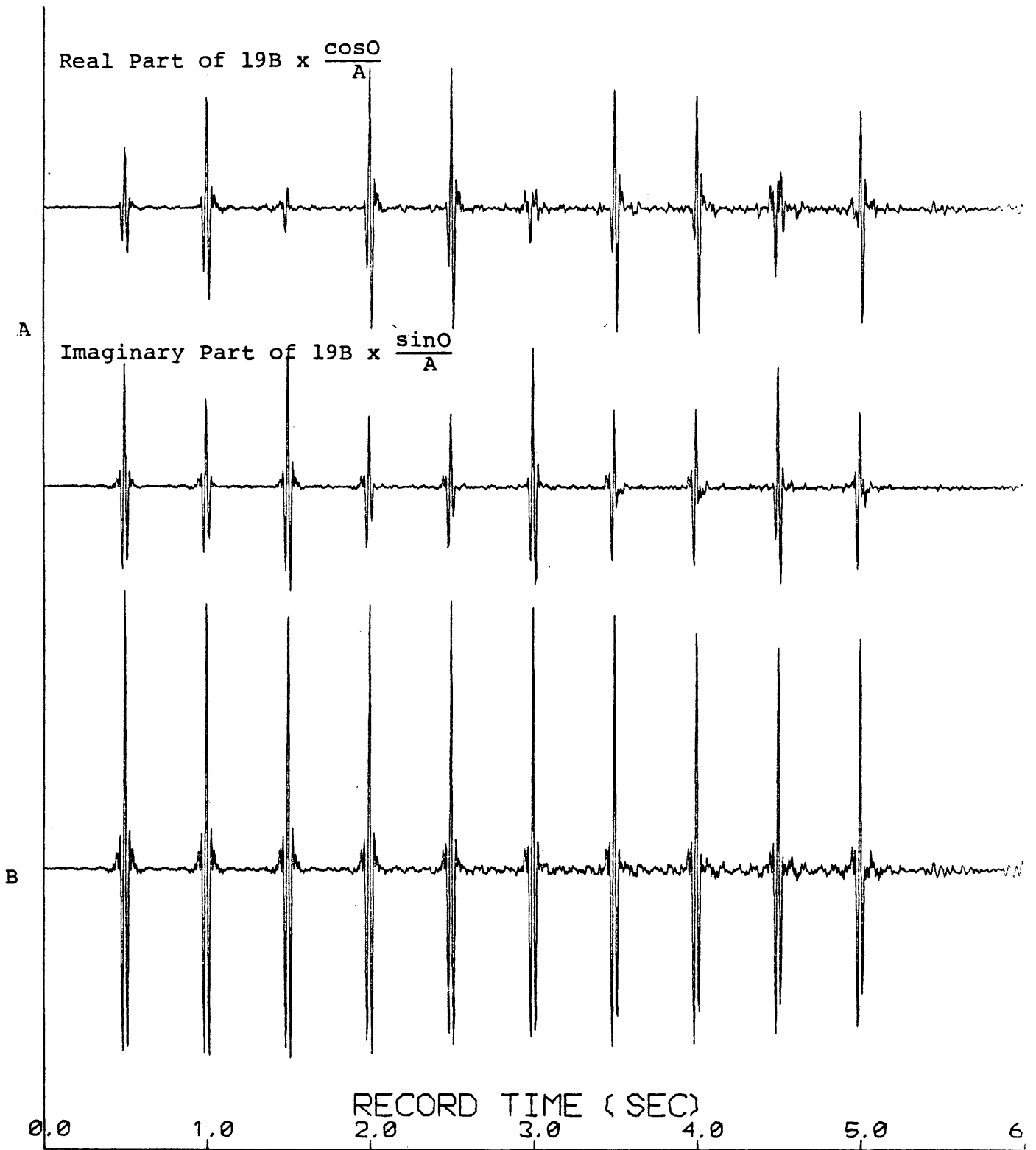


Figure 20A Real and imaginary parts of 19B multiplied by their respective weighting functions
20B Sum of real and imaginary parts in 20A (x2)

Most of the problem of defining the wavelet after transformation is in accounting for the filter effects (amplitude and phase). Figures 21 through 24 are the same as figures 17 through 20, but with no filter. In each plot the constant difference frequencies and the varying sum frequencies were sampled without filtering. Though there is not much character to plots 21A and 23A, the FFT can differentiate the difference frequencies supported throughout the sweep. Note that in figures 21B and 23B the amplitudes of the real and imaginary parts are only weighted by $\cos(aT^2)$ and $\sin(aT^2)$ respectively.

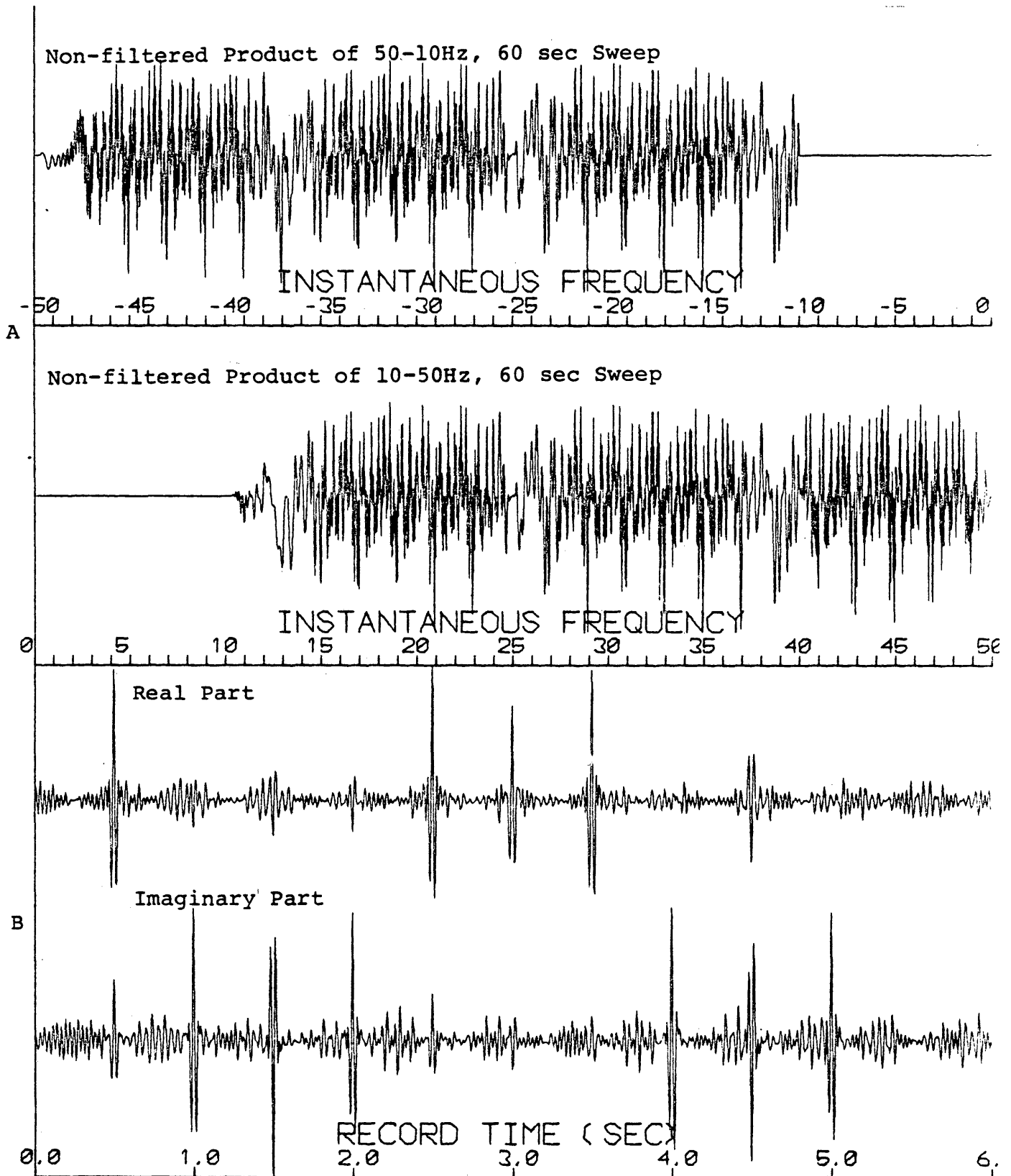


Figure 21A Non-filtered product of 50-10 and 10-50Hz, 60 sec sweeps with 10 reflections at .5 sec intervals from .5 sec to 5.0 sec
21B Real and imaginary parts of Fourier transform of 21A

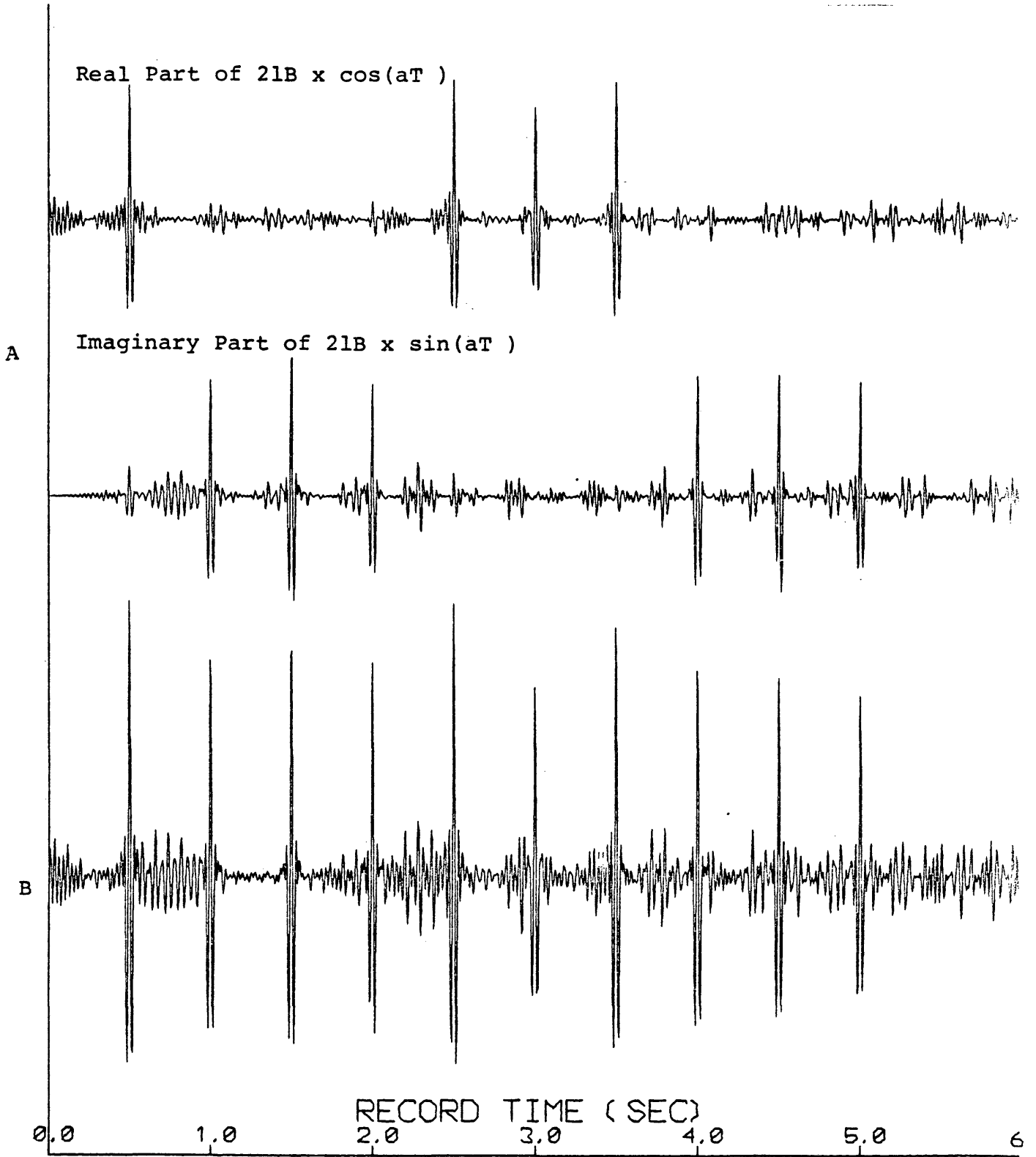


Figure 22A Real and imaginary parts of 2lB multiplied by their respective weighting functions
22B Sum of real and imaginary parts in 22A (x2)

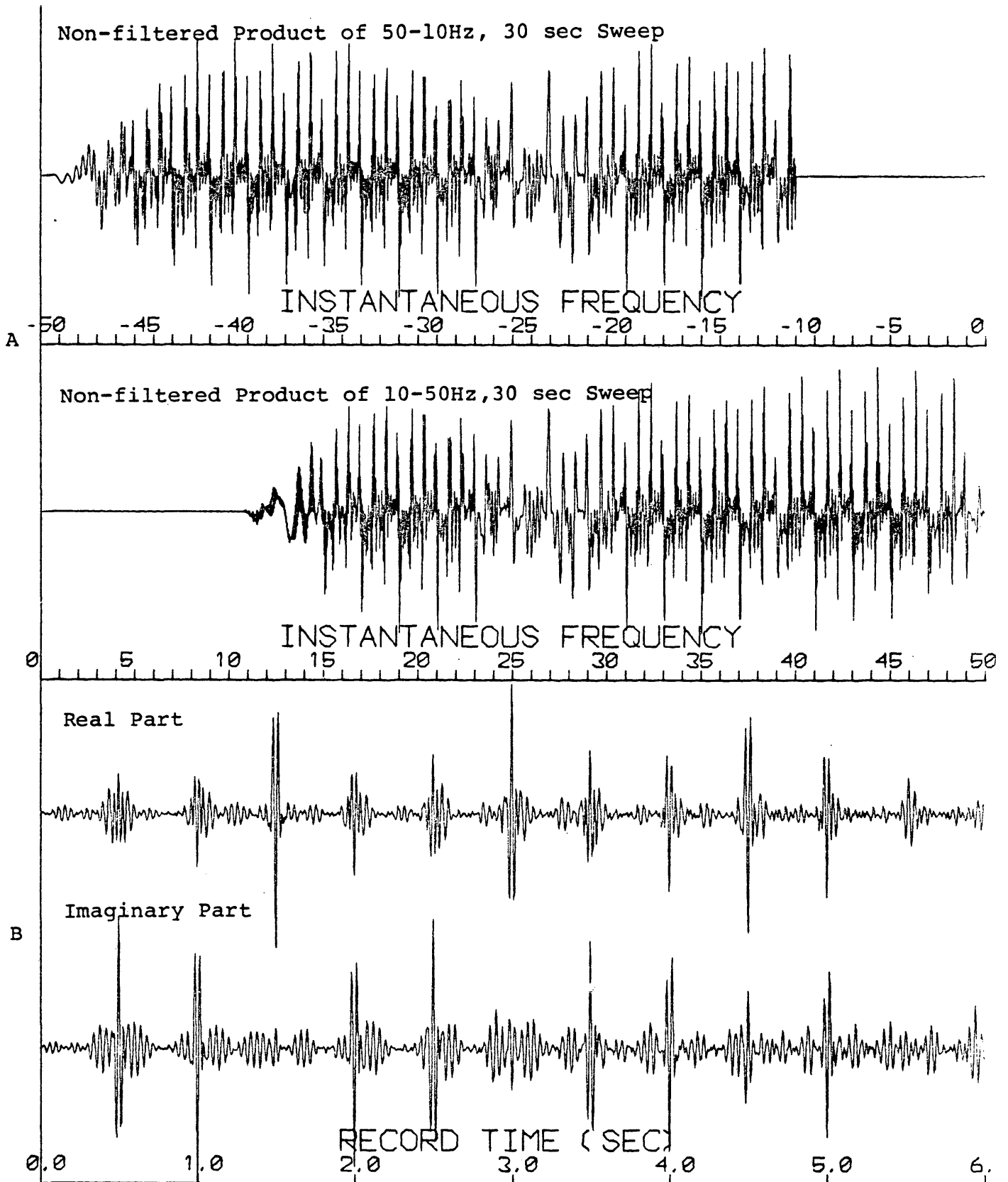


Figure 23A Non-filtered product of 50-10 and 10-50Hz, 30 sec sweeps with 10 reflections at .5 sec intervals from .5 sec to 5.0 sec
23B Real and imaginary parts of Fourier transform of 23A

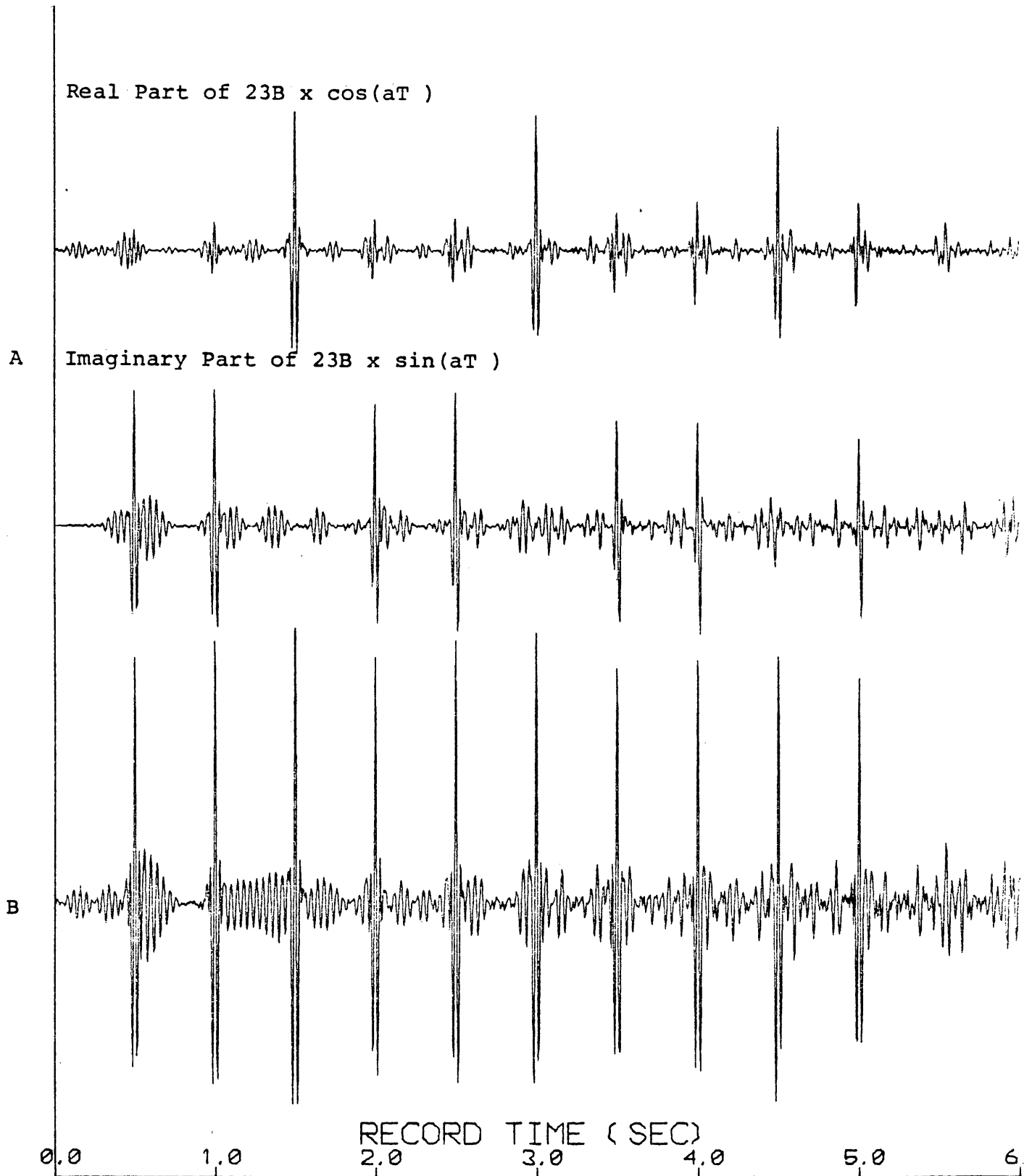


Figure 24A Real and imaginary parts of 23B multiplied by their respective weighting functions
24B Sum of real and imaginary parts in 24A (x2)

The contributions between the major pulses are from aliased sum frequencies. Looking at the product as a function of time (t), a 40ms sample rate will Fourier transform to a normal frequency scale with a 12.5Hz nyquist frequency. The spectrum can be as high as 100Hz so aliasing is quite severe.

As a function of instantaneous frequency (q), the product inverse Fourier transforms to a distribution of record time (T). This distribution will have a folding point at 18.75 sec and content out to 150 sec. Again the aliasing is quite severe.

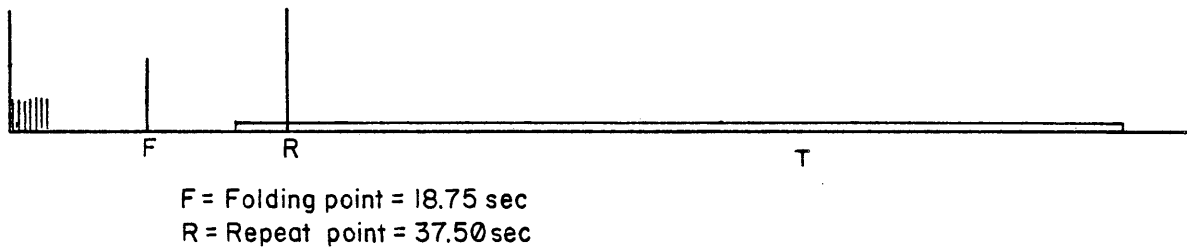


Figure 25

Distribution of amplitude along record time
from 10-50Hz, 60 sec sweep sampled at 40ms.

If the product of this seismic record had been sampled at 8ms the nyquist frequency would be 62.5Hz and the folding point on the distribution of record time would be 93.75 sec. The limit to the frequency spectrum (100Hz) will transform to a wavelet at $T=150$ sec. The 56.25 sec of aliased time will not interfere with the first 37.50 sec of record time.

The result of a non-filtered seismic record from a 10-50Hz, 60 sec sweep, whose product was sampled at 8ms and inverse Fourier transformed, is a record that displays no aliased contributions within the times of interest.

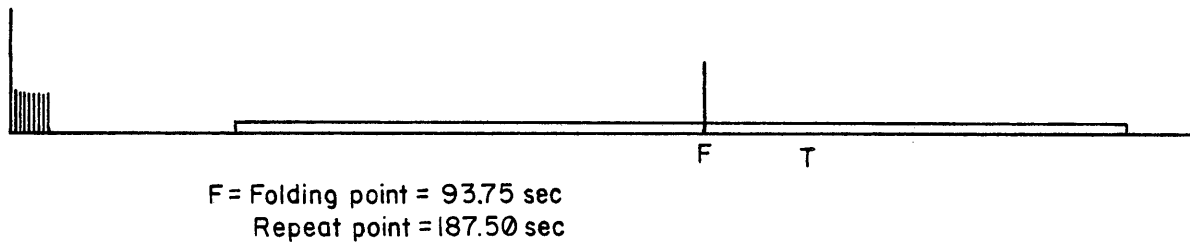


Figure 26

Distribution of amplitude along record time
from 10-50Hz, 60 sec sweep sampled at 8ms.

However, a 60 sec record sampled at 8ms would have 9375 data points to transform. Twice the power of two (for the up sweep and the down sweep) over 9375 is 2^{15} (=32768). A fast Fourier transform over 2^{15} data points would take quite a lot of computing power.

In comparison with the Vibroseis[®] technique VIBROLOKATOR has a lot going for it. A 12 sec. sweep with 5 sec. listening interval, sampled at 4ms, has a recorded signal of 4250 data points to be correlated with a sweep of 3000 data points. The most efficient way to do this is with the FFT. To do this, first one array must be reversed and both augmented with zeros to a power of two. This power of two must be greater than the sum of the data points in each array. This would be 2^{13} (=8192) data points for each. Then the discrete transform of each array is computed.

After transforming, the array that was not reversed must be conjugated and the product of both computed. The correlated record is realized by computing the inverse transform of the product. This is done by conjugating the product and taking the forward transform.

This has involved 3 fast Fourier transforms, (to a power of two equal to 8192), two conjugations and a product.

With the VIBROLOKATOR two 60 second sweeps whose products were sampled at 40ms have 1500 data points each to transform. We must insert 375 zeros to account for the truncated zero to 10Hz in both the up and down sweep, then augment each array from 50Hz to the closest power of two (2048) with zeros. Loading the up sweep in the real part of a complex array and loading the down sweep as the folded half of the array we have an array of 4096 complex points to

[®] Trademark of Continental Oil Company

transform. (In the computer models each sweep was augmented with zeros from 50Hz to 4096 to give twice the data points to transform and twice the definition of the reflection record). After transforming the real array must be multiplied by the cosine and inverse amplitude weighting factors and the imaginary array by the sine and inverse amplitude weighting factors. This need only be done to that number of transformed data points which include the times of interest. And then both arrays are summed. The total computing time for this, even with twice the power of two needed, is less than 1/3 the time involved for correlating the 12 sec. sweep.

The main asset to the VIBROLOKATOR method is the signal to noise ratio that can be realized. Before processing

$$(S/N)_i = B/\sqrt{2CM}$$

where B is the amplitude of the sweep, C is the amplitude of the noise and M is the bandwidth of the noise. After processing

$$(S/N)_o = B\sqrt{T}/\sqrt{C}$$

where T is the sweep length. The signal to noise improvement is

$$(S/N)_o/(S/N)_i = \sqrt{2TW} \sqrt{M/W}$$

where W is the bandwidth of the signal. If the bandwidth of the noise is the same as the bandwidth of the signal, then we have simply, the signal to noise improvement by processing (Harman, 1963).

$$= \sqrt{2TW} .$$

For a 12 sec. 10 to 50Hz sweep, after correlation

$$\sqrt{2TW} = 31.0 = 29.8\text{db} .$$

For two 60 sec. sweeps, 10 to 50Hz and 50 to 10Hz, after transforming

$$\sqrt{2TW} = 97.98 = 39.8\text{db} .$$

This is an improvement of 10db. It takes ten summed and correlated 12 sec. sweeps to equal the signal to noise ratio of one 60 sec. up and one 60 sec. down sweep, Fourier transformed together. The point here is not that you can attain a higher signal to noise ratio with the VIBROKATOR II method (equal amounts of sweep time produce the same ratio), but the ease with which it is attained.

CONCLUSIONS

The geophysics industry has long been one of evolution. Different, and many times better, ways of "doing the same thing" are constantly being introduced. The VIBROLOKATOR method of interpreting vibrator energy sources should be looked at carefully.

It seems that a good signal to noise ratio can be realized with considerably less hardware and software. Complete field processing would be easier. The physical aspects of a slower sweep are less wear on the vibrator and better coupling to the ground. Under conditions in which a relatively long sweep is needed, such as shear wave generation, this method may be truly advantageous.

One pitfall is economy of sweep time. It is many times useful in filtering out ground roll to vibrate a number of times at each station in a centered weighted array. This may be impractical with the VIBROLOKATOR method due to the amount of time spent at each station.

There are many areas in which research and experimentation should be done, such as filter and noise studies. A better low pass filter could make a 30 sec. sweep more reliable. If the non-filtered product of the 60 sec. sweep was sampled at 8ms and transformed, none of the aliased sum frequencies would appear in the first six

seconds of record time.

Other areas of study could include resolution and dynamic range. Resolution would depend upon the bandwidth of the input signal and the ability of the fast Fourier transform to differentiate slight changes in frequency. The difference in signals from the top and bottom of a 100 foot section could be only .007Hz. This would be the case for a 10-50Hz, 60 sec sweep if the section velocity was 10,000 ft/sec. If the same sweep was 30 sec long the difference in signals would be .0134Hz. Intuition says that the dynamic range of this method would be a substantial improvement over conventional methods.

VIBROLOKATOR II--A VIBRATORY SEISMIC METHOD

J. E. White and George Handley
Colorado School of Mines, Golden, Colorado 80401

1. Introduction

Seismic prospecting has always been characterized by changing technology, though the objective of displaying reflected amplitude versus travel time has remained. Many alternative ways of "doing the same thing" have been developed. Innovations may offer lower cost, better quality, environmental acceptability, or simply convenience. The willingness to evaluate so many proposed techniques may be prompted by the thought that any new method may be the preferred method for some set of field conditions and political or environmental constraints.

Vibratory sources have achieved wide acceptance in seismic prospecting. Insofar as correlation of each signal with a sweep pattern is required, use of vibratory sources places certain demands on field recording and subsequent digital processing, in order to display reflections versus time. Described here is a method of deriving the reflection record without correlation, offering potential economies in field data requirements and in processing. These extensions of the technique proposed by I. S. Chichinin will be referred to as Vibrolokator II.

2. Vibrolokator

In attempting to deduce the details of the Vibroseis method (Crawford, 1960) from the sketchy information available to him in Siberia in 1963, Chichinin conceived an alternative method of treating vibratory data so as to produce seismic reflection records (Chichinin, 1974). He proposed the name Vibrolokator for this technique.

The starting-point is a vibrator radiating a sweep signal:

$$b(t) = B \cos at^2 \quad (2-1)$$

T-2082

The instantaneous frequency varies linearly with time:

$$\Omega = \frac{d}{dt} (\alpha t^2) = 2\alpha t \quad (2-2)$$

Discrete reflections can be specified by reflection coefficients h_k and two-way times t_k , giving rise to delayed sweep signals:

$$a(t) = B h_k \cos \alpha (t - t_k)^2 \quad (2-3)$$

Vibrokolator calls for the product of the sweep and the delayed reflections:

$$\begin{aligned} c(t) &= a(t) b(t) = B^2 h_k \cos \alpha (t - t_k)^2 \\ &= (B^2 h_k / 2) [\cos(2\alpha t_k t - \alpha t_k^2) + \cos(2\alpha t^2 - 2\alpha t_k t + \alpha t_k^2)] \end{aligned} \quad (2-4)$$

The instantaneous frequency of the first term is $2\alpha t_k$ and that of the second term $2\alpha(2t - t_k)$. It is assumed that t is large enough that the second term can be filtered out without affecting the first substantially. It is also assumed that α can be made small enough that αt_k^2 can be neglected. Then the filtered product trace is

$$\bar{c}(t) = (B^2 h_k / 2) \cos(2\alpha t_k t) \quad (2-5)$$

If many discrete reflectors are present

$$\bar{c}(t) = (B^2 / 2) \sum_{k=0}^{\infty} h_k \cos(2\alpha t t_k) \quad (2-6)$$

We may consider reflection coefficient to be a continuous function of time T (replacing t_k), in which case

$$\bar{c}(t) = (B^2 / 2) \int_0^{\infty} h(T) \cos(2\alpha t T) dT \quad (2-7)$$

With $\Omega = 2\alpha t$, the filtered product $\bar{c}(\Omega)$ is seen to be the real part of the Fourier transform of the reflectivity function $h(T)$. Since

the reflectivity function is causal, only the real part is needed. The reverse transform yields

$$h(T) = \int_0^{\infty} \bar{c}(\Omega) \cos(\Omega T) d\Omega.$$

3. Vibrolokator II

An extension of the Vibrolokator approach as described below will be referred to as Vibrolokator II. The steps required to preserve waveform and amplitude of all reflections, regardless of record time, will be set forth in detail.

The starting-point is a linear sweep varying as $B \cos \alpha t^2$, an example of which is shown in Figure 1. A typical reflection of strength h_k at time t_k contributes a received signal proportional to $B h_k \cos \alpha (t - t_k)^2$. The product of sweep and signal is, as before:

$$c(t) = (B^2 h_k / 2) [\cos(2\alpha t_k t - \alpha t_k^2) + \cos(2\alpha t^2 - 2\alpha t_k t + \alpha t_k^2)] \quad (3-1)$$

This is a particular function of time t , shown in Figure 2, which has a Fourier transform in terms of angular frequency ω :

$$C(\omega) = \int_{-\infty}^{\infty} c(t) e^{-i\omega t} dt \quad (3-2)$$

The first term in Eq. (3-1) contributes delta-functions at $\omega = \pm 2\alpha t_k$ to $C(\omega)$. The second term has instantaneous frequency $(4\alpha t - 2\alpha t_k)$ ranging from $-\infty$ to ∞ , being zero at $t = t_k/2$. This suggests that its contribution to $C(\omega)$ contains all frequencies, including the neighborhood of $2\alpha t_k$. If the function $c(t)$ is limited to a window $w(t)$ such that $|t| \gg t_k$, then the spectral contribution of the second term is concentrated in a frequency range well above $2\alpha t_k$ and hence can be filtered out by a low-pass filter. The validity of this idea can be appreciated by examining Figure 3.

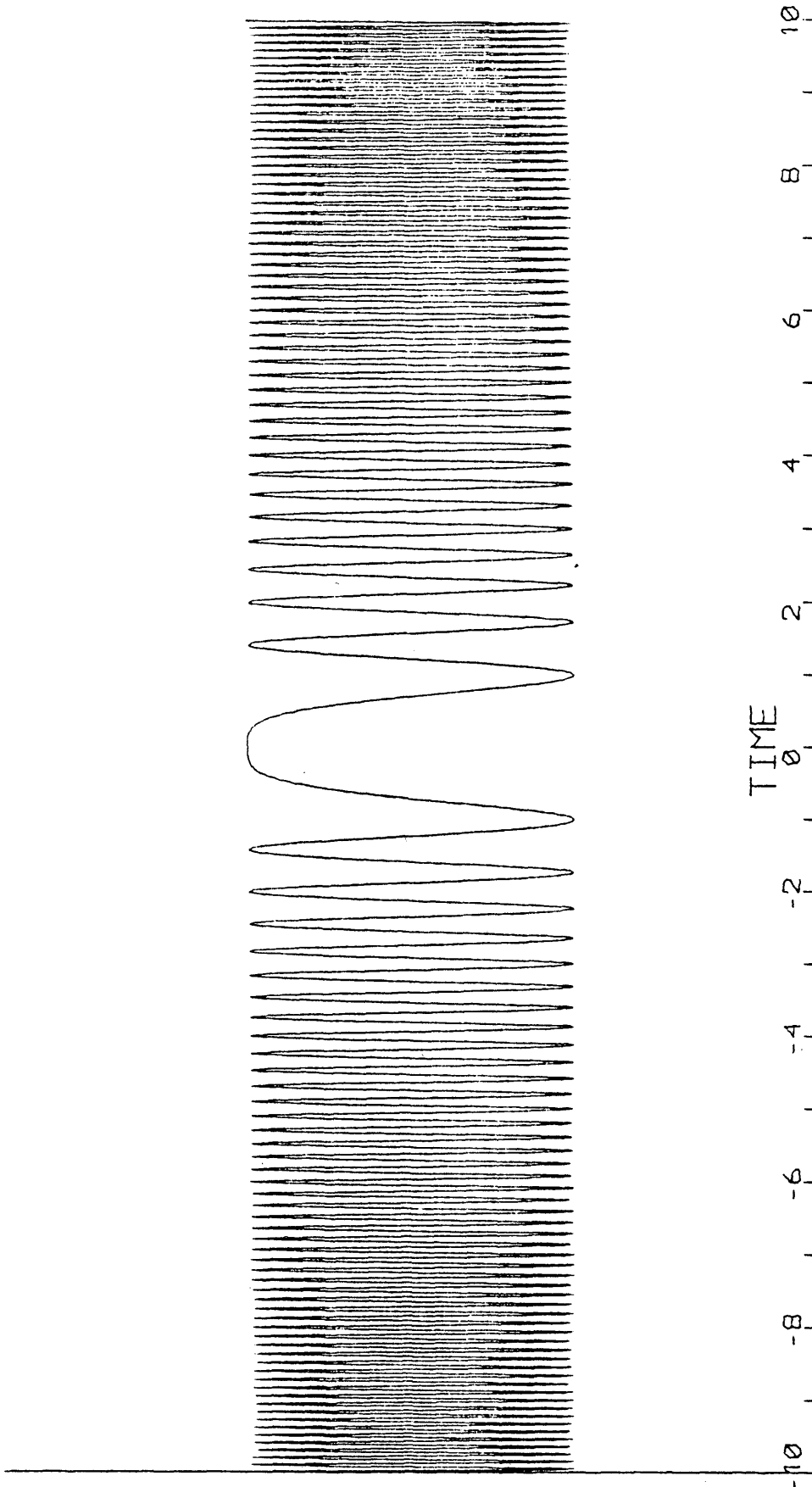


FIGURE 1. SWEEP FROM - 10 Hz TO + 10 Hz IN 20 SECONDS

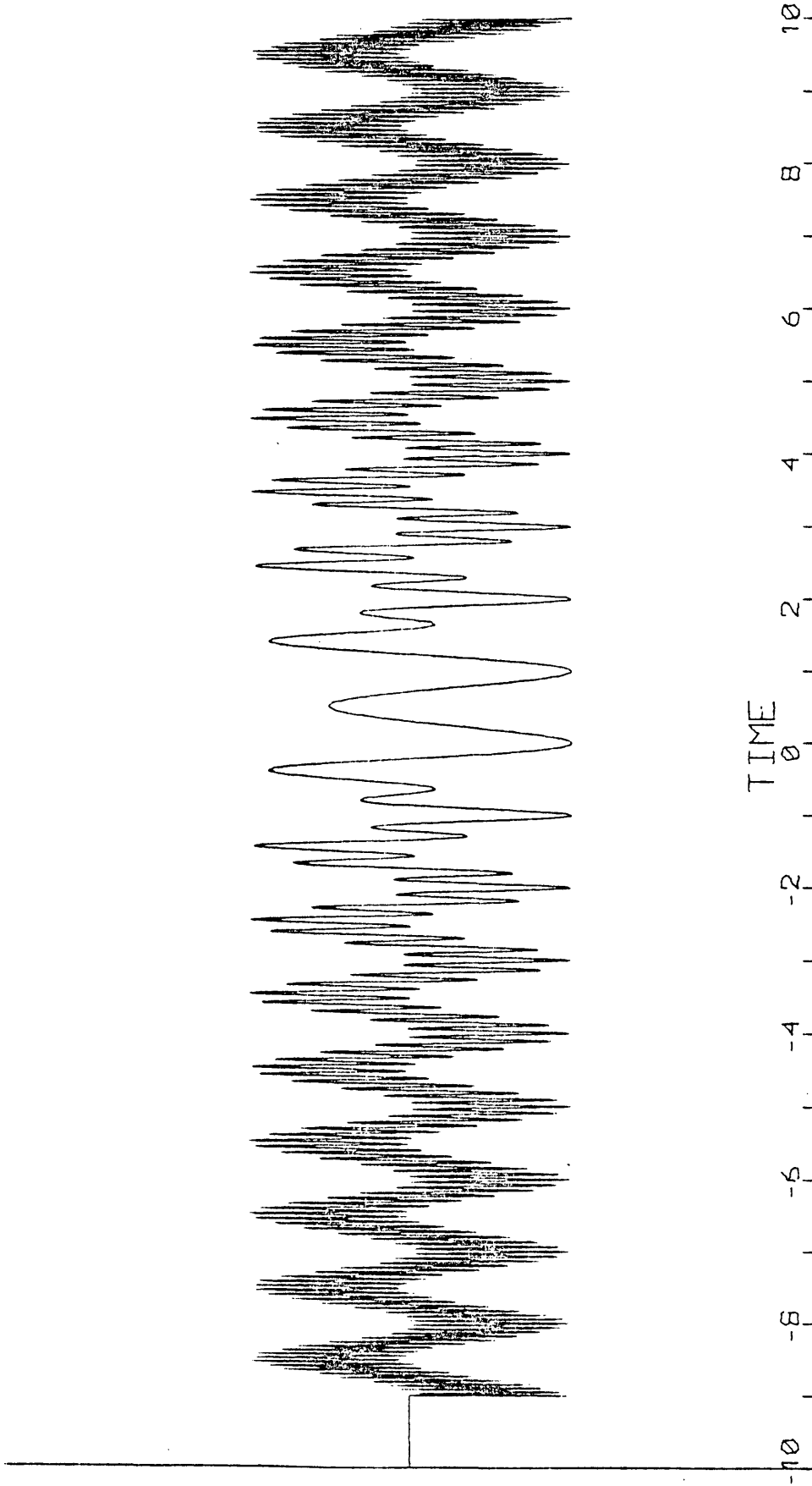


FIGURE 2. PRODUCT OF SWEEP AND ONE-SECOND REFLECTION

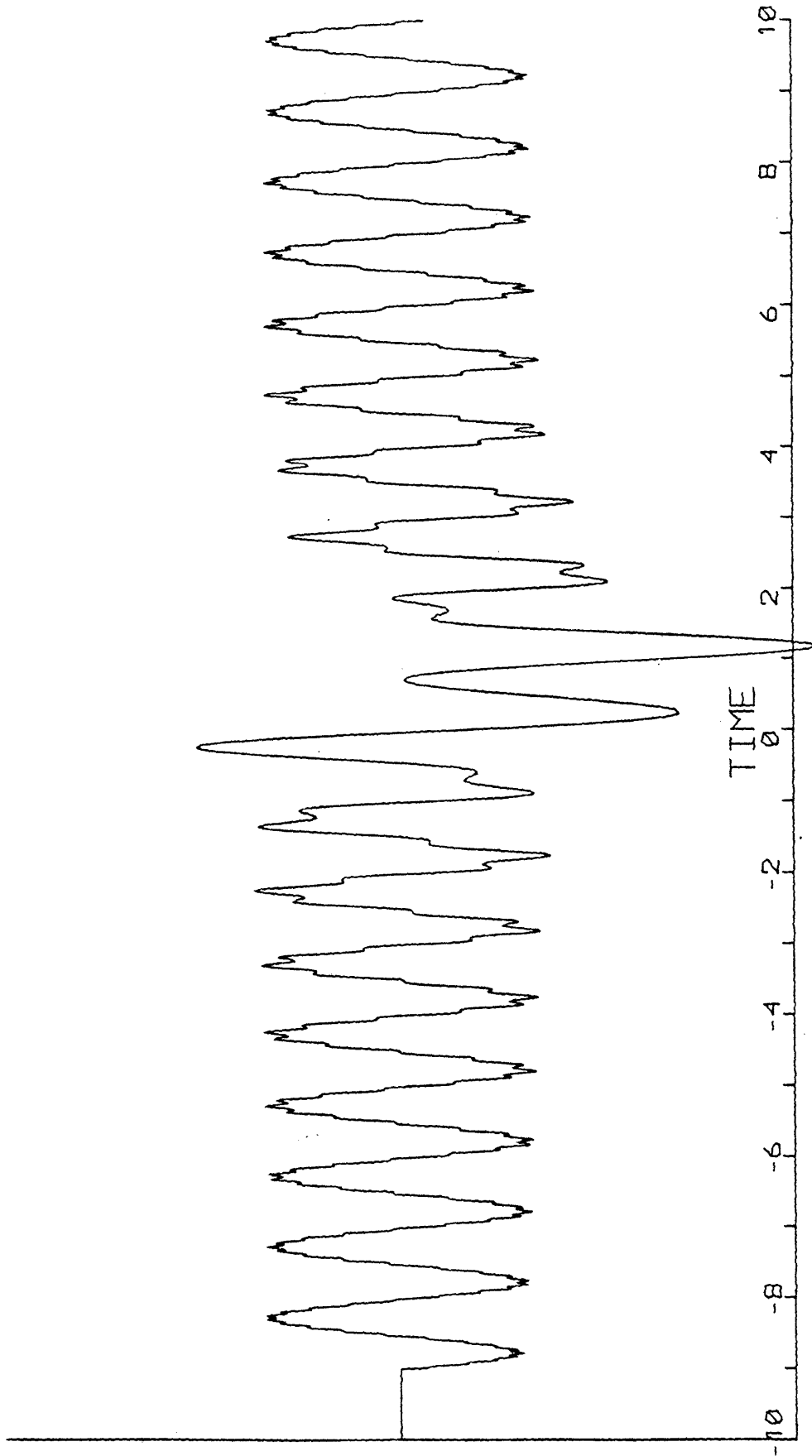


FIGURE 3. PRODUCT AFTER SINGLE-SECTION LOW-PASS FILTER

The product trace of Figure 2 has been filtered digitally with the equivalent of a single section of an RC low-pass filter, and sinusoidal waveform is almost free of the high-frequency jitter at either end of the plot, near 10 Hz. A window passing -50 to -10 Hz and 10 to 50 Hz would cut out all the distorted waveform shown in Figure 3 and hence would consist of virtually a sine wave, for each reflection. In any case, we assume that the filter reduces the second term to a negligible contribution.

We express the filter in terms of amplitude and phase $A(\omega) e^{i\theta(\omega)}$ and designate $A(2\alpha t_k)$ as A_k and $\theta(2\alpha t_k)$ as θ_k . Then the filtered and window-limited product is:

$$\bar{c}(t) = (B^2 h_k / 2) w(t) A_k \cos(2\alpha t_k t - \alpha t_k^2 - \theta_k) \quad (3-3)$$

For convenience in recognizing the Fourier transform of this expression, use the definition $q = 2\alpha t$ and express the window $w(t)$ in terms of this new variable. The transform of the window itself is

$$W(T) = \frac{1}{2\pi} \int_{-\infty}^{\infty} w(q) e^{iqT} dq \quad (3-4)$$

The real variable T has the units of time. With ϕ_k defined as $(\alpha t_k^2 + \theta_k)$, the transform of the filtered product is

$$C(T) = \frac{1}{2\pi} \int_{-\infty}^{\infty} (B^2 h_k A_k / 2) w(q) \cos(q t_k - \phi_k) e^{iqT} dq \quad (3-5)$$

Expanding the cosine term into exponentials, we see that its transform consists of delta-functions:

$$\begin{aligned} & \frac{1}{2\pi} \int_{-\infty}^{\infty} (1/2) (e^{iqt_k} e^{-i\phi_k} + e^{-iqt_k} e^{i\phi_k}) e^{iqT} dq \\ &= (1/2) [\delta(T + t_k) e^{-i\phi_k} + \delta(T - t_k) e^{i\phi_k}] \\ &= (1/2) [\delta(T - t_k) + \delta(T + t_k)] \cos \phi_k \\ & \quad + (i/2) [\delta(T - t_k) - \delta(T + t_k)] \sin \phi_k \end{aligned} \quad (3-6)$$

The multiplication by $w(q)$ in Eq. (3-5) implies that the terms in Eq. (3-6) are to be convolved with the basic wavelet $W(T)$, determined by the nature of the window. Thus the first term in the real part of Eq. (3-6) places a wavelet at $T = t_k$, multiplied by the reflection coefficient h_k , a filter factor A_k , and $\cos\theta_k$. The imaginary part contributes a wavelet at $T = t_k$, multiplied by h_k , by A_k , and by $\sin\theta_k$. This is expressed as

$$C(T) = (B^2/4) h_k A_k [W(T - t_k) \cos\theta_k + iW(T - t_k) \sin\theta_k] \quad (3-7)$$

for $T > 0$

An example of this result is shown in Figure 4, for a sweep covering 10-50 Hz in 60 seconds. Ten reflections of equal amplitude are assumed, in the first five seconds. The upper curve (real part of Eq. 3-7) shows six reflections of negative polarity and four of positive polarity, with widely varying amplitudes. The lower curve (imaginary part) is similar. The parameter B , which depends on vibrator design and earth coupling, may be hard to evaluate in practice. For this analysis, it has been taken to be constant. The ratio $A(2\alpha t_k)/A(2\alpha T)$ is unity when $T = t_k$. If the pulse $W(T - t_k)$ is short enough that it is essentially zero except in the neighborhood of t_k , this ratio will compensate for the effect of the filter on this typical reflection. Similarly, $\cos[\alpha T^2 + \theta(2\alpha T)]$ and $\cos[\alpha t_k^2 + \theta(2\alpha t_k)]$ are equal at $T = t_k$ and approximately equal in the neighborhood of t_k . Two new functions are derived from Eq. (3-7):

$$D(T) = h_k [A(2\alpha t_k)/A(2\alpha T)] W(T - t_k) \cos[\alpha t_k^2 + \theta(2\alpha t_k)] \cos[\alpha T^2 + \theta(2\alpha T)] \quad (3-8)$$

$$E(T) = h_k [A(2\alpha t_k)/A(2\alpha T)] W(T - t_k) \sin[\alpha t_k^2 + \theta(2\alpha t_k)] \sin[\alpha T^2 + \theta(2\alpha T)]$$

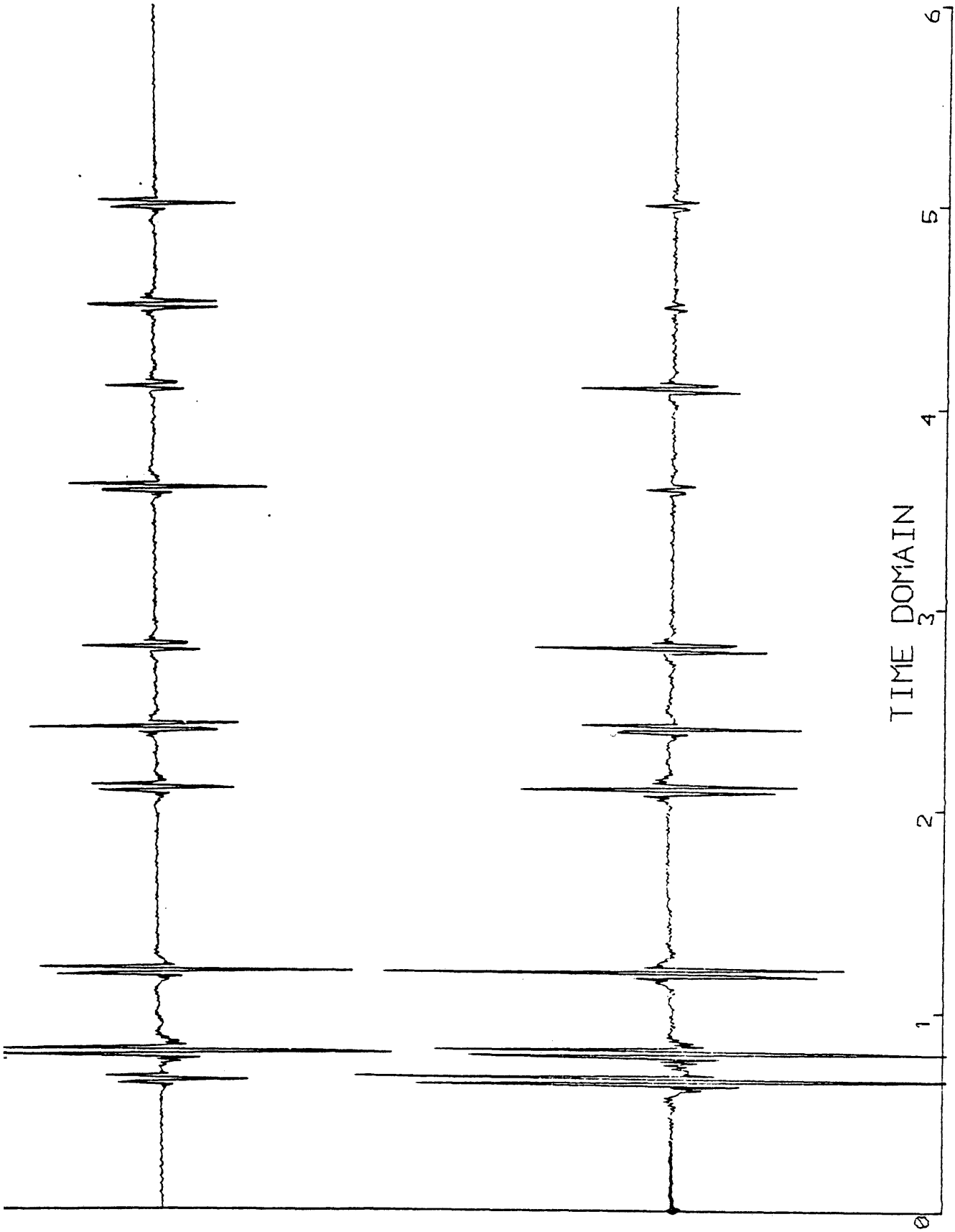


FIGURE 4. TRANSFORM OF THE FILTERED PRODUCT. UPPER CURVE, REAL PART OF EQ. 3-7. LOWER CURVE, IMAGINARY PART.

An example of these two functions is shown in Figure 5, derived from the curves of Figure 4. Note that all reflections in $D(T)$, shown in the upper curve, have positive sense. All reflections in $E(T)$ also have the same sense (Negative, because this curve was inadvertently inverted). The sum of these two gives the reflection wavelet at time t_k , proportional to the reflection coefficient h_k :

$$R_k(T) = h_k W(T - t_k) \quad (3-9)$$

The reflection record is the sum of contributions from K reflectors:

$$R(T) = \sum_{k=1}^K h_k W(T - t_k) \quad (3-10)$$

The example of this is shown in Figure 6. All reflections have positive polarity and approximately equal amplitudes. If reflectivity is considered to be a continuous function of two-way time τ , then the record trace becomes the convolution of this function $h(\tau)$ with the wavelet:

$$R(T) = \int_{-\infty}^{\infty} h(\tau) W(T - \tau) d\tau \quad (3-11)$$

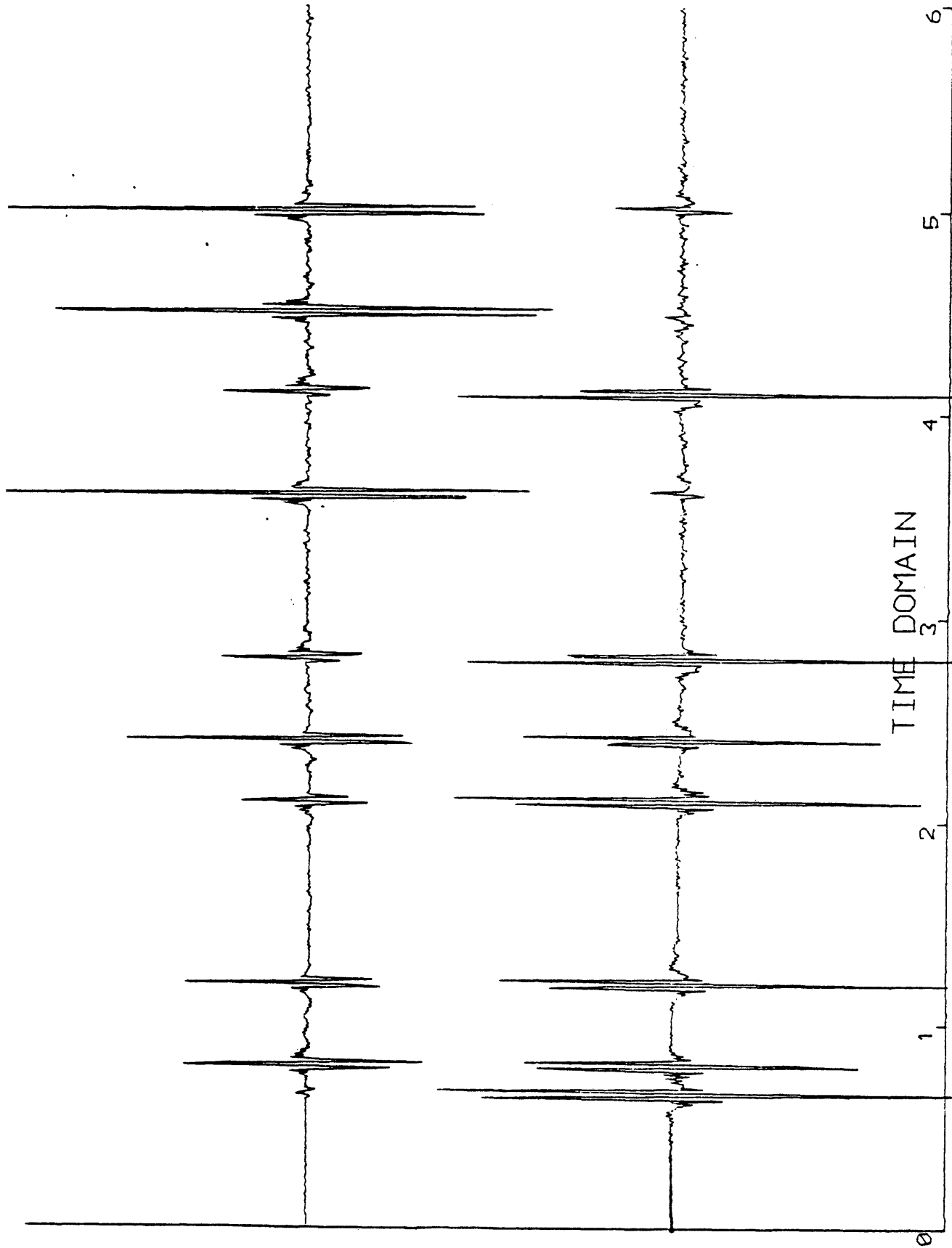


FIGURE 5. TRANSFORM CORRECTED ACCORDING TO EQUATION 3-8.
UPPER CURVE, (D), LOWER CURVE, $-E(T)$.

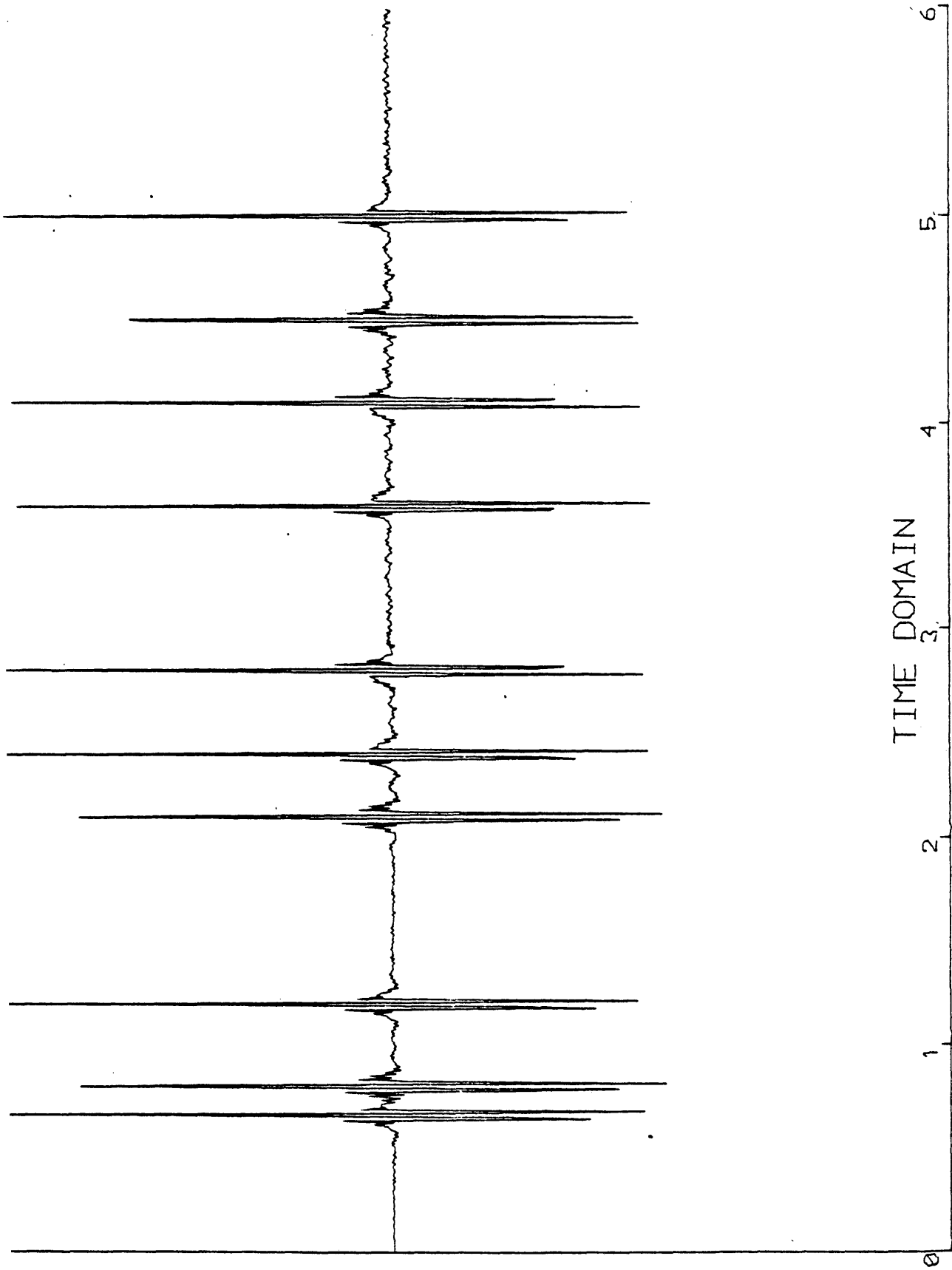


FIGURE 6. TRANSFORM OF PRODUCT TRACE WITH ALL CORRECTIONS APPLIED

LIST OF SYMBOLS

- q: a real variable
t: real time during frequency sweep
 t_k : two-way time for kth reflection
 τ : continuous two-way time for reflection
T: record time
 ω : angular frequency for spectrum of any $f(t)$
 Ω : instantaneous frequency of sweep, $2\alpha t$

References

1974,
Chichinin, I. S., Institute of Geology and Geophysics,
Siberian Branch of USSR Academy of Sciences, Novosibirsk
(Personal communication)

Crawford, John M., William E. N. Doty and Milford R. Lee,
1960, Continuous Signal Seismograph, Geophysics, vol. 25,
pp. 95-105.

USERS MANUAL

Introduction

All computer work was done on a DEC SYSTEM 10 MODEL 1055 computer.

This is a manual to explain the nine programs used in generating and analyzing data for both the VIBROLOKATOR and VIBROLOKATOR II methods. Included is a listing of each program. The explanation of each program is complete in terms of use, but a basic knowledge of fortran will be needed to completely understand them. Each program is divided into sections that are noted on the program in the form of comments. The program explanations in this users manual are written by section, for each program. Each explanation includes a generalized flow chart, a list of the core requirements, a description of each question (if any) asked by the program with an example, an estimate of the CPU time required to execute the program, and the command used to execute the program.

These programs are interactive and the variables in each are defined the same. A list of these is included before the program explanations.

The programs and some of the data are stored on a DEC Tape. A list of what is stored and an explanation of how to access the tape is also included.

Contents

Page 68	Introduction
Page 71	Stored Programs and Data
Page 73	List of Variables
Page 76	VIB10.FOR
Page 85	VIBRO1.FOR
Page 88	PLOTV1.FOR
Page 91	VIBRO2.FOR
Page 95	PLOTV2.FOR
Page 98	PLOTV3.FOR
Page 100	NOFL10.FOR
Page 100	NOFILT.FOR
Page 101	ANALOG.FOR

Stored Programs and Data

The programs are all stored on a DEC tape. To mount the tape, type

```
MOU DTA:VIB/V:DE334-VIBRO/WL
```

To copy from the tape to your area, type

```
COPY=VIB:File1,File2,.....etc.
```

To dismount the tape type DISMOU VIB:

In addition, stored on this tape is the data taken with the analog model,

DATA9.DAT is the first reading

DATA10.DAT is the second reading

DATA11.DAT is the third reading

and some data generated from VIB10.FOR.

FOR18.DAT=Parameters for 600 sec up sweep.

FOR19.DAT=600 sec up sweep w reflections every .5 sec

FOR20.DAT=Parameters for 30 sec up and down sweep

FOR21.DAT=30 sec up sweep w reflections every .5 sec

FOR22.DAT=30 sec down sweep w reflections every .5 sec
FOR23.DAT=Parameters for 60 sec up and down sweep
FOR24.DAT=60 sec up sweep w reflections every .5 sec
FOR25.DAT=60 sec down sweep w reflections every .5 sec
FOR26.DAT=60 sec up sweep w reflections every .1 sec
FOR27.DAT=60 sec down sweep w reflections every .1 sec

The parameter files should be input as FOR07.DAT to the FFT programs (VIBRO1.FOR or VIBRO2.FOR). The up and down sweeps should be input as FOR08.DAT and FOR09.DAT respectively.

List of Variables

This is a list of all the variables, determined by the input parameters. The variables and how they are determined are in capitals. An explanation of each follows. They are listed in the order that they appear.

$LAST = SPS * TIME$ Last is the number of data points in the sweep.

$GAP = LOW * LAST / (HI - LOW)$ Gap is the number of data points that are truncated from zero Hz to LOW.

$CT = CT * SPS$ This redefines the coupling time in terms of data points.

$TOTAL = GAP + LAST$ This is the total number of data points from zero Hz to HI.

$TOTIME = TIME + GAP / SPS$ This is the total time from zero Hz to HI.

$END = TOTAL / SR$ This is the total number of data points in the sampled product array.

$START = GAP / SR$ This is the number of data points in the sampled array that represent the truncated section from zero Hz to LOW.

$ALPHA = PI*HI/TOTIME$ (PI=3.141592654) This is pi times the rate of change of sweep.

$A = ALPHA/(SPS**2)$ This redefines alpha so that we can multiply by integers in the do loop.

$FILTCO = -ALOG(FC)*SPS$ This is equal to "k" in the text.

$M =$ the second power of two greater than end. This determines the duration of the fourier transform in the next program.

$NDATA = 2**M$ This is the power of two that the fast Fourier transform will operate over.

$NALIAS = NDATA/2$ This is the alias point or the last point in the first half of the array.

$REPFRE = NDATA*HI/END$ This is the frequency limit in the array to be transformed.

$REPTIM = END/HI$ This is the time at which the transformed array will begin to repeat itself.

$A1 = ALPHA/(REPFRE**2)$ This redefines alpha so that we can multiply by integers when determining the phase angle added by the reflection (A1T).

$TANG = 2*ALPHA*REPTIM/(NDATA*FILTCO)$ This is the tangent of the phase angle added by the filter.

$W = (2 * \text{ALPHA} * \text{REPTIM} / \text{NDATA}) ** 2$ This is the square of the customary angular frequency that the filter is acting on.

$F = \text{FILTCO} ** 2$ This is the square of the filter coefficient (k).

$\text{PHI} = \text{ATAN}(\text{TANG} * (\text{I}-1))$ This is the phase angle at each point that is added by the filter.

$\text{AMP} = (F + W * (\text{I}-1) ** 2) ** .5$ This is the inverse of the filter amplitude.

UP This is the name given to the up-sweep array.

DN This is the name given to the down-sweep array.

RE This is the name given to the array that contains the real part of the Fourier transform.

AI This is the name given to the array that contains the imaginary part of the Fourier transform.

SUM This is the name given to the array that contains the sum of the real and imaginary parts of the Fourier transform after they have been multiplied by their respective weighting functions. When plotted, this normalized array is multiplied by two.

VIB10.FOR

This program generates a filtered sampled product of an up sweep and a down sweep that is analogous to the VIBROKATOR and VIBROKATOR II methods of recording field data. The output from this program can be input into VIBRO1.FOR for an analysis of the data using the VIBROKATOR method or input into VIBRO2.FOR for an analysis of the data using the VIBROKATOR II method.

The only core requirements for this program are for the arrays in which the up sweep (UP) and down sweep (DN) are to be written. Both are dimensioned at 1876. The limiting factors here are the future programs. To execute VIBRO2.FOR the UP and DN arrays together must be less than 4000.

The program is broken into eight sections for easy reference in this discussion. See page 83 for the flow chart and page 107 for the source code.

Section 1

Seventeen questions are asked by the program. These questions either define or are used to define all constants and variables needed by the program. The answers to these questions may be typed in integer or floating point format. Here is a list of the questions with an explanation. The questions are capitalized, and an example of a typical answer is given for each. In parentheses after each

question is the constant used to identify it in the program.

INITIAL FREQUENCY = This is the frequency at which you would like to start your pilot sweep. Ex. = 10 (LOW)

FINAL FREQUENCY = This is the frequency at which you would like to end your pilot sweep. Ex. = 50 (HI)

TIME = This is the length (in sec) of your sweep. Ex. = 60 (TIME)

SAMPLES PER SECOND = This is how fast you want your data sampled while generating it. Ex. = 250 (SPS)

FIRST REFLECTION TIME = This is the time (in sec) at which your first reflection is received and begins to be multiplied with the pilot sweep. Ex. = .68 (T1)

The next nine questions are the same as the last, concerning reflection times 2 thru 10. (T2 - T10)

COUPLING TIME = This is the time (in sec) it takes the pilot sweep to linearly increase in amplitude from zero to one. Ex. = 1 (CT)

FILTER CONSTANT = This is the amount of the last data value added to the present data value by the recursive filter. The recursion equation is $Y = X_n + FC(Y_{n-1})$. Ex. = .99 (FC)

SAMPLE RATE = This is the rate at which the filtered product is sampled. This must be an integer. If you type in 10, it means that every tenth data value of the filtered sampled product will be stored. In other words, if you generated the data at a 4ms sample rate you are sampling the product at 40ms. Ex. = 10 (SR)

Section 2

Additional variables needed by the program (see Variables List) are computed and all variables needed by future programs are written to FOR07.DAT.

Section 3

Each reflection time is redefined in terms of data points ($T1 = T1 * SPS$). The starting time in data points for each reflection from zero is determined ($D1 = GAP + T1$) and the ending time in data points ($E1 = TOTAL + T1$).

Section 4

The frequency arrays created here are to be inverse Fourier transformed. The transformation requires that the arrays begin at zero Hz and increase linearly to some upper limit. The fast Fourier transform program requires that the upper limit be represented by a data point that is the Mth data point in which M is a power of two. This section inserts zeros into the UP and DN arrays to correspond to the truncated section from zero Hz to LOW. Before transforming

(in VIBRO1.FOR or VIBRO2.FOR) zeros will be inserted into the array to transform to correspond to the truncated section from HI to the folding point (section 3 of VIBRO1.FOR and section 4 of VIBRO2.FOR).

Section 5

Section 5 initializes indices for section 6. "H" is used with the index (I) to generate the down sweep. As the index increases, the down sweep, a function of H-I, decreases. "J" is the index for the sampled arrays that are output. "K" is the counter for decimation after filtering. Each time through the do loop it is increased by one and tested to be equal to the sample rate. If it is, the value of the array at that point is stored and K is set equal to zero. "Y" and "S" are the values of the data after being filtered. They are assigned an initial value of zero.

Section 6

The do loop in which the data is generated runs from I = GAP to TOTAL. The up sweep is straight forward. For each index I,

U = I Redefine the index as floating point because integer format does not have enough accuracy to square the large values of I.

$$UP1 = \text{COS}(A*(U**2)) \quad \text{Pilot sweep}$$

$URF1 = \text{COS}(A*((U-T1)**2))$ First up reflection,
 equal to the pilot delayed by T1 (is zero if I is less than
 D1).

$X = UP1*(URF1+....+URF10)$ Sum reflections and
 multiply with pilot.

$$Y = X + FC*Y \quad \text{Filter}$$

Each time through the do loop, a counter is increased
 by one and tested to be equal to the sample rate (SR). If
 it is then the value from the filter equation (Y) is stored
 in UP and the counter goes back to one.

The down sweep is generated at the same time but
 running from the high frequency to the low frequency. This
 is done by first defining $H = \text{TOTAL} + \text{GAP}$. When we redefine
 the index as floating point we define it as $D = H - I$. From
 here on the down sweep is generated the same as the up,
 replacing all the U's with D's. In this case when the
 first reflection is the pilot delayed by T1, the equation is

$DRF1 = \text{COS}(A*((D+T1)**2))$ Now the reflection is coming
 in at a higher instantaneous frequency (earlier time) than
 the pilot.

The summed product of the down sweep is R and the
 output to the filter is S.

In each pilot and reflection the sweep has a linear increase and decrease in amplitude determined by the coupling time at the beginning and the end. For the start of the sweep this is done by redefining the sweep if I is less than the starting time of the sweep plus CT.

$$URF1 = URF1*((I-D1)/CT)$$

For the end of the sweep this is done by redefining the the sweep if I is greater than the ending time of the sweep minus CT.

$$URF1 = URF1*((E1-I)/CT)$$

Section 7

After generating the data, the down sweep array, from end to start, is reversed in order. Now the down sweep array varies in frequency just as the up sweep array. There are zeros from zero Hz to start, and then data from start to end that represents the filtered sampled product of frequencies from LOW to HI.

Section 8 The UP array is then written to FOR08.DAT and the DN array is written to FOR09.DAT.

To execute this program type

EX VIB10.FOR

For a 30 sec sweep generated at a 4ms sample rate this program requires about 70 sec of CPU time. For a 60 sec sweep generated at 4ms, it requires about 140 sec.

VIB10.FCR

Section 1

Section 2

Section 3

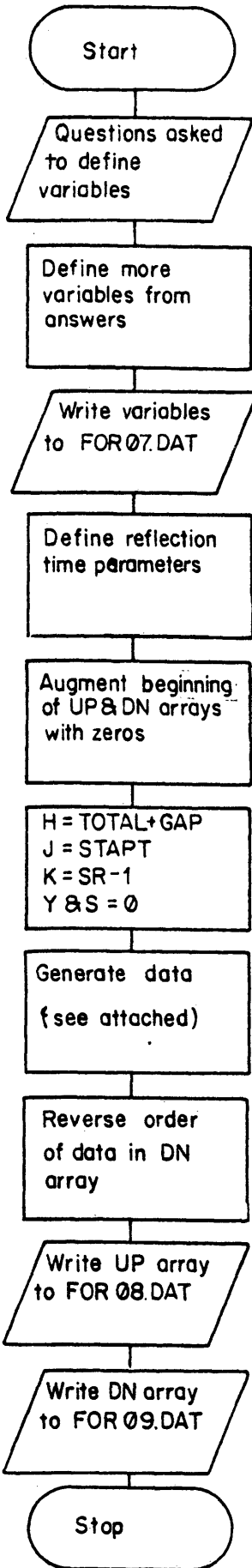
Section 4

Section 5

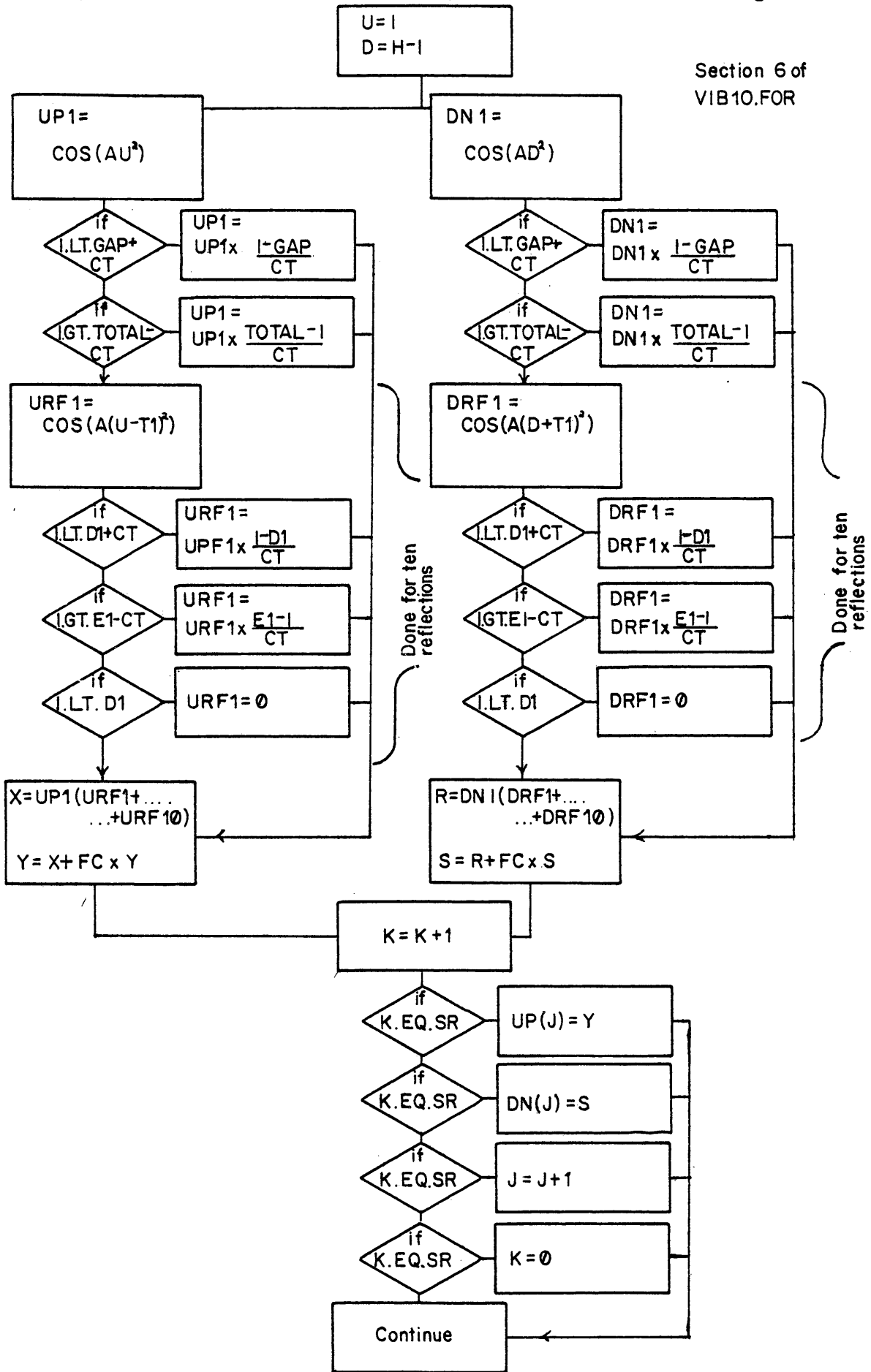
Section 6

Section 7

Section 8



Section 6 of
VIB10.FOR



VIBRO1.FOR

This program inverse Fourier transforms the up sweep written to FOR08.DAT from the program VIB10.FOR. The input to this program and the output can be plotted by the program PLOTV1.FOR (see Figures 11-13, pages 23-25)..

This program uses a lot of core. The FFT power of two is 13. This is twice what is needed but gives twice the definition to the wavelet. Since the array SWEEP is complex the actual dimension is twice 8192. The program needs a work area (IWK) dimensioned at one more than the complex array (8193). So with the UP array, the total core requirements for dimensioning is 26,453.

The program is broken into five sections for easy reference in this discussion. See page 87 for the flow chart and page 113 for the source code.

Section 1

The variables needed by the program are read from FOR07.DAT and additional variables are defined or redefined. The up sweep is read from FOR08.DAT.

Section 2

The up sweep is multiplied with a Hanning truncator.

Section 3

Now the UP array is loaded into the real part of the complex array SWEEP. The imaginary part is assigned a value of zero. The data is loaded to END and then the array is augmented with zeros from END+1 to the folding point. The folding point NFOLD is assigned a value of zero and then the first half of the array is folded into the second half.

Section 4

The resulting array is inverse fast Fourier transformed using the IMSL library programs FFT2 and FFRDR2. For an explanation of these programs refer to the library documentation (good luck).

Section 5

After transformation there is only a real part (the imaginary part is all zero). The real part is written to FOR14.DAT, from which it can be plotted along with the up sweep by PLOTV1.FOR.

To execute this program type

```
EX VIBOR1.FOR,LBY:IMSL/LIB
```

This program requires about 20 sec of CPU time.

VIBRO1.FOR

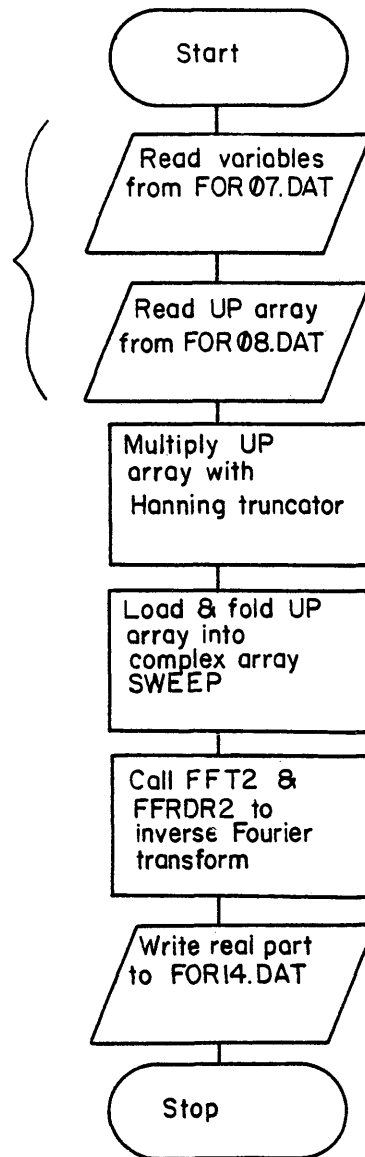
Section 1

Section 2

Section 3

Section 4

Section 5



PLOTV1.FOR

This program plots the filtered sampled product of the up sweep from VIB10.FOR and the real part of the Fourier transform from VIBR01.FOR.

The core requirements of this program are not very great. Just enough to read the arrays and print the axis titles. The axis titles and the plot dimensions are pre-set within the program.

The program is divided into five sections for easy reference to this discussion. See page 90 for the flow chart and page 115 for the source code.

Section 1

The variables needed by the program are read from FOR07.DAT. From these additional variables are defined (see Variable List). One question is asked to determine the amount of data (seconds) you want on the transformed axis. The answer to this question is used to redefine NDATA. The up sweep is read from FOR08.DAT and the real part of the transformed array is read from FOR14.DAT.

Section 2

Here the UP array is tested and normalized. This is done so that when the plot routine is called there will be no scaling problem.

Section 3

The RE array is now tested and normalized as before.

Section 4

The plot routine is called, first to plot the axis with title, and then to plot the UP array.

Section 5

The plot routine is again called, this time for the RE axis and title, and then the RE array.

To execute this program type

EX PLOTV1.FOR

This program requires about 12 sec of CPU time to execute. This time amount includes plotting all of the up-sweep and six seconds of the real part of the transformed data.

PLOTV1.FOR

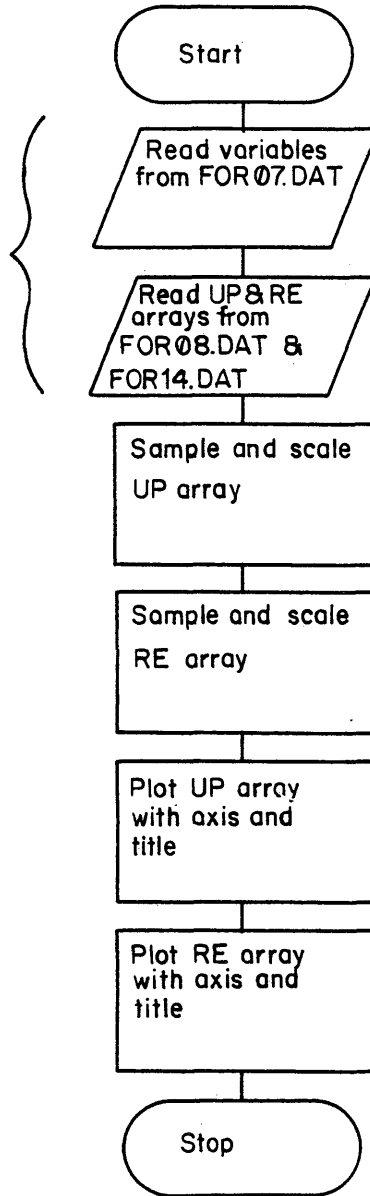
Section 1

Section 2

Section 3

Section 4

Section 5



VIBRO2.FOR

This program computes the inverse fast Fourier transform of the up sweep - down sweep combination from VIB10.FOR. Such a combination is shown in Figure 15A, page 33. There are four outputs from this program. The real and imaginary parts of the transformed array are written to FOR14.DAT and FOR15.DAT respectively. These can be plotted with the up sweep - down sweep combination by executing PLOTV2.FOR after this program (see Figure 15, page 33). Also output are the real and imaginary parts after having been multiplied by the cosine and sine weighting functions. These are written to FOR16.DAT and FOR17.DAT respectively and can be plotted along with their sum by executing PLOTV3.FOR after this program (see Figure 16, page 34).

This program uses a lot of core. The FFT power of two is 13. This is twice what is needed but gives twice the definition to the wavelet. Since the array SWEEP is complex the actual dimension is twice 8192. The program needs a work area (IWK) dimensioned at one more than the complex array (8193). So with the UP and DN arrays the total core requirements for dimensioning are 28,329.

This program is divided into seven sections for easy reference to this discussion. See page 94 for the flow chart and page 118 for the source code.

Section 1

The variables needed by the program are read from FOR07.DAT. Additional variables, including those that define the phase angle and filter amplitude, are calculated from those read (see Variables List).

Section 2

The filtered decimated up sweep (UP) and down sweep (DN) are read from FOR08.DAT and FOR09.DAT respectively.

Section 3

Both arrays are multiplied with a Hanning truncator.

Section 4

The up sweep is loaded into the real part of the first half of the complex array sweep. The imaginary part is all zero and the array is augmented with zeros from END+1 to NALIAS. The folding point NFOLD is assigned a value of zero. Then the down sweep is loaded into the folded half of SWEEP (NFOLD+1 to NDATA). This is done by loading DN from 1 to END in SWEEP, from NDATA backwards. The array is augmented with zeros from NDATA-END to NFOLD+1. The result is a real array that is symmetric in instantaneous frequency scale.

Section 5

The complex array SWEEP is inverse fast Fourier transformed using the IMSL library programs FFT2 and FFRDR2. For an explanation of these programs refer to the library documentation.

Section 6

The real and imaginary parts of the transformed array are written to FOR14.DAT and FOR15.DAT respectively. Executing PLOTV2.FOR after this program plots the real and imaginary parts with the up and down sweep.

Section 7

The real and imaginary parts, after being transformed are weighted by a filter amplitude function and a phase angle function. Each part is now multiplied by the inverse of the filter amplitude function and by its respective phase angle function. The approximate result is both parts weighted only by the square of the phase angle function. These new arrays are written to FOR16.DAT and FOR17.DAT respectively.

To execute this program type

```
EX VIBRO2.FOR,LBY:IMSL/LIB
```

This program requires about 30 sec of CPU time.

VIBRO2.FOR

Section 1

Section 2

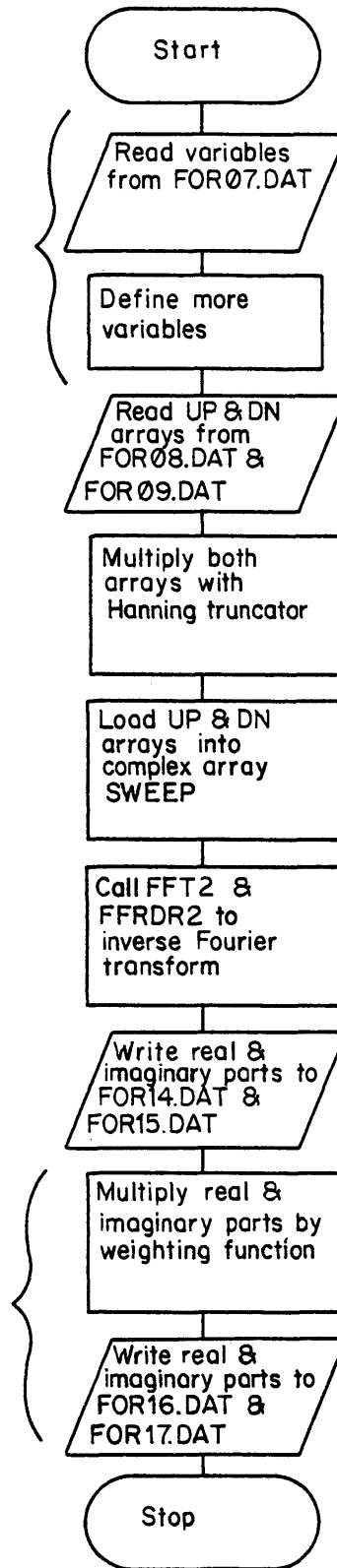
Section 3

Section 4

Section 5

Section 6

Section 7



PLOTV2.FOR

This program plots the products of the up sweep and the down sweep from VIB10.FOR or NOFL10.FOR. It also plots the real and imaginary parts of the transformed array from VIBRO2.FOR or NOFILT.FOR.

The core requirements of this program are not very great. Just enough to read the arrays and print the axis titles. The axis titles and plot dimensions are pre-set within the program.

The program is divided into five sections for easy reference to this discussion. See page 97 for the flow chart and page 121 for the source code.

Section 1

The variables are read from FOR07.DAT and from these additional variables are defined (see Variables List). One question is asked to determine the amount of data (seconds) you want on the transformed axis. The answer to this question is used to redefine NDATA. The up sweep (UP) and down sweep (DN) are read from FOR08.DAT and FOR09.DAT respectively. The real part (RE) and the imaginary part (AI) are read from FOR14.DAT and FOR15.DAT respectively.

Section 2

The UP and DN arrays are tested and normalized independently.

Section 3

The RE and AI arrays are tested and normalized independently.

Section 4

In this section the axis and title for each array is plotted (the real and imaginary plots share the same axis).

Section 5

The plot routines are called and the UP and DN arrays are plotted.

Section 6

The plot routines are called and the RE and AI arrays are plotted.

To execute this program type

EX PLOTV2.FOR

This program requires about 19 sec of CPU time. This time amount includes plotting all of the up and down sweeps and six seconds of the real and imaginary parts of the transformed data.

PLOTV2.FOR

Section 1

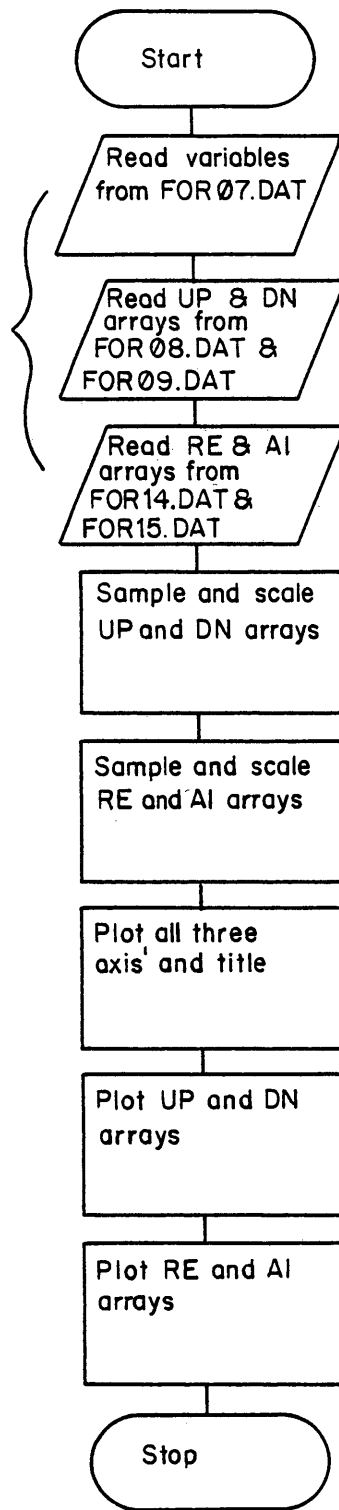
Section 2

Section 3

Section 4

Section 5

Section 6



PLOTV3.FOR

This program plots the real and imaginary parts of the transformed array from VIBRO2.FOR or NOFILT.FOR after they have been multiplied by their respective weighting functions. It also plots the sum of these two parts.

The core requirements of this program are not very great. Just enough to read the arrays, define a SUM array, and print the title. The axis title and plot dimensions are pre-set within the program.

The program is divided into six sections for easy reference to this discussion. See page 100 for the flow chart and page 124 for the source code.

Section 1

The variables needed by this program are read from FOR07.DAT and additional variables are calculated. One question is asked to determine the amount of data (seconds) you want on the axis. The answer to this question is used to redefine NDATA. The real (RE) and imaginary (AI) parts are read from FOR16.DAT and FOR17.DAT respectively.

Section 2

In this section the RE and AI arrays are summed and written to the array SUM.

Section 3

The three arrays (RE, AI, and SUM) are tested and normalized independently.

Section 4

In this section the axis with title is plotted. The three arrays share the same axis.

Section 5

The plot routine is called and the normalized RE and AI arrays are plotted.

Section 6

The plot routine is called and the normalized SUM array is plotted, multiplied by two.

To execute this program type

EX PLOTV3.FOR

This program requires about 9 sec of CPU time. This time amount includes plotting six seconds of the transformed data for each array.

PLOTV3.FOR

Section 1

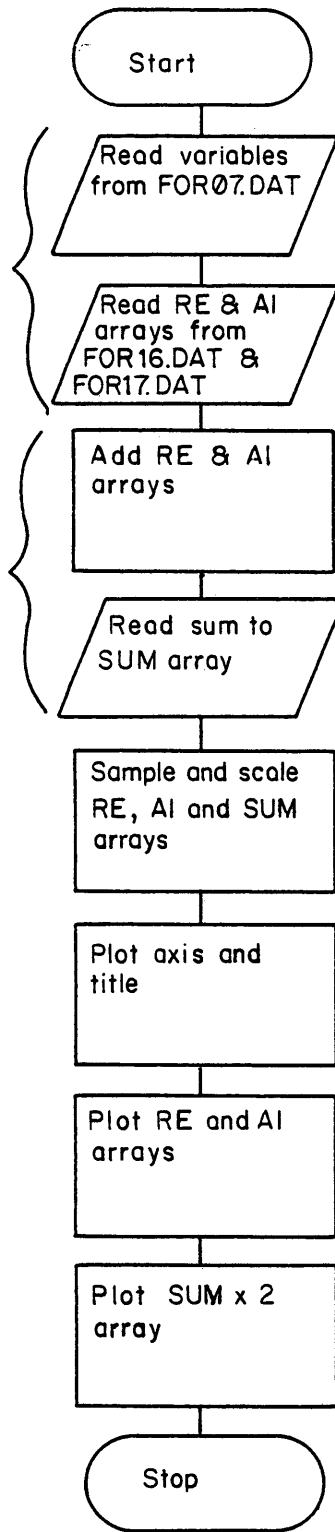
Section 2

Section 3

Section 4

Section 5

Section 6



NOFL10.FOR

This program is exactly the same as VIB10.FOR except it has no filter. See page 127 for the source code. To execute this program type:

EX NOFL10.FOR

NOFILT.FOR

This program is exactly the same as VIBRO2.FOR except that it has no filter effects to account for (amplitude and phase). The phase angle added to the wavelet by the reflection is squared as before. See page 133 for the source code. To execute this program type:

EX NOFILT.FOR,LBY:IMSL/LIB

ANALOG.FOR

This program is the same, basically, as VIBRO1.FOR except that the input is one of the three data files taken during the three analog readings. The file that is input is inverse fast Fourier transformed using the IMSL library programs FFT2 and FFRDR2. To transform one of the three files it must first be copied to DATA.DAT. Not much core is needed as the FFT power of two is 10.

The program is divided into eight sections for easy reference in this discussion. See page 105 for the flow chart and page 136 for the source code.

Section 1

Four questions are asked about the data, from which all variables are defined.

FREQUENCY INCREMENT = This is the sample rate in frequency (Hz) at which the data was read. Ex = 25 (IFRE)

INITIAL FREQUENCY = This is the frequency at which the readings were started. Ex = 500 (IFRE1)

FINAL FREQUENCY = This is the frequency at which the readings were finished. Ex = 2500 (IFREF)

POWER OF TWO = This is the number of points over which the FFT will be taken. Ex = 9 (I)

Section 2

The complex array CDATA is augmented with zeros from it's begining to the point at which the data is to start. This sequence of zeros correspond to the truncated section at the begining of the data.

Section 3

The data copied to DATA.DAT is loaded into the complex array CDATA.

Section 4

The complex array CDATA is augmented with zeros from the final data point to the power of two used by the FFT.

Section 5

The IMSL programs FFT2 and FFRDR2 are called to inverse fast Fourier transform the array CDATA.

Section 6

The real and imaginary parts of the transformed array are written to FOR02.DAT, complete with column headings and a time index to identify your position in record time.

Section 7

The real part of the transformed array is tested and normalized.

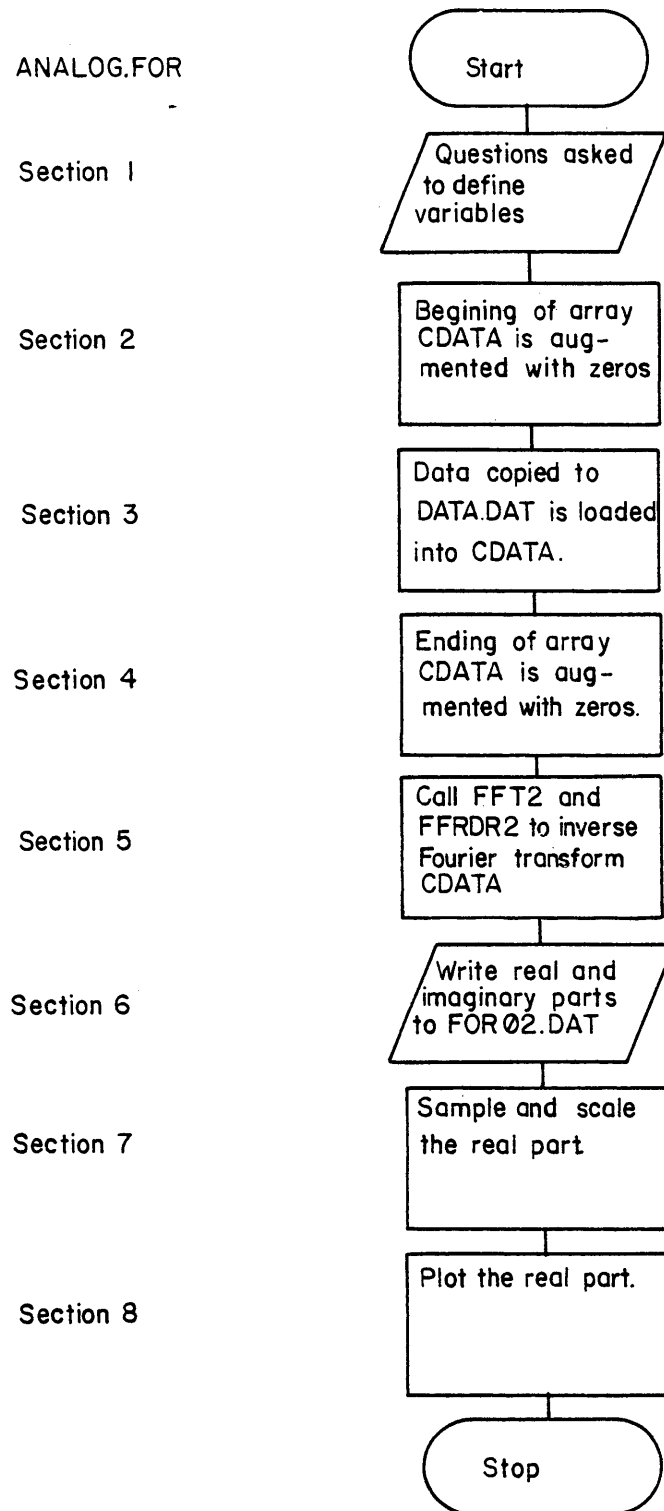
Section 8

The real part of the transformed array is plotted.

To execute this program type

EX ANALOG.FOR,LBY:IMSL/LIB

This program requires about 18 sec of CPU time to execute.



PROGRAMS

Below is a listing of each program. Comments (identified with a "C" at the beginning of the line) have been inserted into each program. These are ignored by the program and are there solely to help identify the different parts of each program.

Page 107	VIB10.FOR
Page 113	VIBRO1.FOR
Page 115	PLOTV1.FOR
Page 118	VIBRO2.FOR
Page 121	PLOTV2.FOR
Page 124	PLOTV3.FOR
Page 127	NOFL10.FOR
Page 133	NOFILT.FOR
Page 136	ANALOG.FOR

C -----VIB10.FOR-----

C George Handley May 1978

C This program generates a filtered sampled product
 C of an up sweep (UP) and a down sweep (DN)
 C that is analogous to the VIBROLOKATOR and VIBROLOKATOR II
 C methods of recording field data. The output to this can
 C be input into VIBRO1.FOR for an analysis of the
 C data using the VIBROLOKATOR method or input
 C into VIBRO2.FOR for an analysis of the data using the
 C VIBROLOKATOR II method.

C Seventeen questions are asked by this program.
 C These questions either define or are used to define all
 C constants and variables needed by the program. The answers
 C to these questions may be typed in integer or floating
 C point format. The output to this program is
 C written to three data files for use in future programs.
 C All constants needed to define future constants and
 C variables are written to FOR07.DAT. The filtered sampled
 C up sweep is written to FOR08.DAT. The filtered sampled
 C down sweep is written to FOR09.DAT. This program is
 C divided into eight sections. A comment before each section
 C briefly identifies it. For a more complete description
 C refer to the users manual for thesis # T-2082, on file
 C with the Geophysics Department, Colorado School of Mines.

```
DIMENSION UP(0/1875),DN(0/1875)
REAL LOW
INTEGER END
PI=3.141592654
```

C Section 1

C In this section all questions are asked to determine the
 C variables needed by the program.

```
WRITE(4,1)
1  FORMAT(" INITIAL FREQUENCY=", $)
  READ(4,2)LOW
2  FORMAT(F)
  WRITE(4,3)
3  FORMAT(" FINAL FREQUENCY=", $)
  READ(4,2)HI
  WRITE(4,4)
4  FORMAT(" TIME=", $)
  READ(4,2)TIME
  WRITE(4,5)
5  FORMAT(" SAMPLES PER SECOND=", $)
  READ(4,2)SPS
```

```

        WRITE(4,6)
6       FORMAT(" FIRST REFLECTION TIME=", $)
        READ(4,2)T1
        WRITE(4,7)
7       FORMAT(" SECOND REFLECTION TIME=", $)
        READ(4,2)T2
        WRITE(4,8)
8       FORMAT(" THIRD REFLECTION TIME=", $)
        READ(4,2)T3
        WRITE(4,40)
40      FORMAT(" FOURTH REFLECTION TIME=", $)
        READ(4,2)T4
        WRITE(4,41)
41      FORMAT(" FIFTH REFLECTION TIME=", $)
        READ(4,2)T5
        WRITE(4,42)
42      FORMAT(" SIXTH REFLECTION TIME=", $)
        READ(4,2)T6
        WRITE(4,43)
43      FORMAT(" SEVENTH REFLECTION TIME=", $)
        READ(4,2)T7
        WRITE(4,44)
44      FORMAT(" EIGHTH REFLECTION TIME=", $)
        READ(4,2)T8
        WRITE(4,45)
45      FORMAT(" NINTH REFLECTION TIME=", $)
        READ(4,2)T9
        WRITE(4,46)
46      FORMAT(" TENTH REFLECTION TIME=", $)
        READ(4,2)T10
        WRITE(4,9)
9       FORMAT(" COUPLING TIME=", $)
        READ(4,2)CT
        WRITE(4,10)
10      FORMAT(" FILTER CONSTANT=", $)
        READ(4,2)FC
        WRITE(4,11)
11      FORMAT(" SAMPLE RATE=", $)
        READ(4,2)SR

```

C Section 2

C In this section additional variables will be computed
 C from the answers given already, and those needed for future
 C programs will be written to FOR07.DAT.

```

LAST=SPS*TIME
GAP=LOW*LAST/(HI-LOW)
CT=CT*SPS
TOTAL=GAP+LAST
TOTIME=TIME+GAP/SPS
END=TOTAL/SR

```

```

START=GAP/SR
ALPHA=PI*HI/TOTIME
A=ALPHA/(SPS**2)
FILTCO=-ALOG(FC)*SPS
M=1
12  IF(2**M.GT.END)GO TO 13
    M=M+1
    GO TO 12
13  M=M+1
14  FORMAT(G)
    WRITE(7,14)HI
    WRITE(7,14)END
    WRITE(7,14)M
    WRITE(7,14)START
    WRITE(7,14)PI
    WRITE(7,14)ALPHA
    WRITE(7,14)FILTCO

```

C Section 3

C In this section the times governing the parameters of each
C sweep will be defined or redefined in terms of samples
C per second.

```

T1=T1*SPS
T2=T2*SPS
T3=T3*SPS
T4=T4*SPS
T5=T5*SPS
T6=T6*SPS
T7=T7*SPS
T8=T8*SPS
T9=T9*SPS
T10=T10*SPS
  D1=GAP+T1
D2=GAP+T2
D3=GAP+T3
D4=GAP+T4
D5=GAP+T5
D6=GAP+T6
D7=GAP+T7
D8=GAP+T8
D9=GAP+T9
D10=GAP+T10
E1=TOTAL+T1
E2=TOTAL+T2
E3=TOTAL+T3
E4=TOTAL+T4
E5=TOTAL+T5
E6=TOTAL+T6
E7=TOTAL+T7
E8=TOTAL+T8

```

```

E9=TOTAL+T9
E10=TOTAL+T10

```

C Section 4

C This do loop assigns a value of zero to all the points that
C would correspond to the truncated section of the sweep from
C zero Hz to LDW.

```

      DO 19 I=0,START-1
      UP(I)=0
      DN(I)=0
19    CONTINUE

```

C Section 5

C These next five lines define counters needed by the program.

```

H=TOTAL+GAP
J=START
K=SR-1
Y=0
S=0

```

C Section 6

C This do loop generates, sums, multiplies, filters, and
C samples the data. Both for the up sweep and the down sweep.

```

      DO 20 I=GAP,TOTAL
      U=I
      D=H-I
      UP1=COS(A*(U**2))
      IF(I.LT.GAP+CT)UP1=UP1*((I-GAP)/CT)
      IF(I.GT.TOTAL-CT)UP1=UP1*((TOTAL-I)/CT)
      DN1=COS(A*(D**2))
      IF(I.LT.GAP+CT)DN1=DN1*((I-GAP)/CT)
      IF(I.GT.TOTAL-CT)DN1=DN1*((TOTAL-I)/CT)
      URF1=COS(A*((U-T1)**2))
      IF(I.LT.D1+CT)URF1=URF1*((I-D1)/CT)
      IF(I.GT.E1-CT)URF1=URF1*((E1-I)/CT)
      IF(I.LT.D1)URF1=0
      DRF1=COS(A*((D+T1)**2))
      IF(I.LT.D1+CT)DRF1=DRF1*((I-D1)/CT)
      IF(I.GT.E1-CT)DRF1=DRF1*((E1-I)/CT)
      IF(I.LT.D1)DRF1=0
      URF2=COS(A*((U-T2)**2))
      IF(I.LT.D2+CT)URF2=URF2*((I-D2)/CT)
      IF(I.GT.E2-CT)URF2=URF2*((E2-I)/CT)
      IF(I.LT.D2)URF2=0
      DRF2=COS(A*((D+T2)**2))
      IF(I.LT.D2+CT)DRF2=DRF2*((I-D2)/CT)
      IF(I.GT.E2-CT)DRF2=DRF2*((E2-I)/CT)
      IF(I.LT.D2)DRF2=0

```

```

URF3=COS(A*((U-T3)**2))
IF(I.LT.D3+CT)URF3=URF3*((I-D3)/CT)
IF(I.GT.E3-CT)URF3=URF3*((E3-I)/CT)
IF(I.LT.D3)URF3=0
DRF3=COS(A*((D+T3)**2))
IF(I.LT.D3+CT)DRF3=DRF3*((I-D3)/CT)
IF(I.GT.E3-CT)DRF3=DRF3*((E3-I)/CT)
IF(I.LT.D3)DRF3=0
URF4=COS(A*((U-T4)**2))
IF(I.LT.D4+CT)URF4=URF4*((I-D4)/CT)
IF(I.GT.E4-CT)URF4=URF4*((E4-I)/CT)
IF(I.LT.D4)URF4=0
DRF4=COS(A*((D+T4)**2))
IF(I.LT.D4+CT)DRF4=DRF4*((I-D4)/CT)
IF(I.GT.E4-CT)DRF4=DRF4*((E4-I)/CT)
IF(I.LT.D4)DRF4=0
URF5=COS(A*((U-T5)**2))
IF(I.LT.D5+CT)URF5=URF5*((I-D5)/CT)
IF(I.GT.E5-CT)URF5=URF5*((E5-I)/CT)
IF(I.LT.D5)URF5=0
DRF5=COS(A*((D+T5)**2))
IF(I.LT.D5+CT)DRF5=DRF5*((I-D5)/CT)
IF(I.GT.E5-CT)DRF5=DRF5*((E5-I)/CT)
IF(I.LT.D5)DRF5=0
URF6=COS(A*((U-T6)**2))
IF(I.LT.D6+CT)URF6=URF6*((I-D6)/CT)
IF(I.GT.E6-CT)URF6=URF6*((E6-I)/CT)
IF(I.LT.D6)URF6=0
DRF6=COS(A*((D+T6)**2))
IF(I.LT.D6+CT)DRF6=DRF6*((I-D6)/CT)
IF(I.GT.E6-CT)DRF6=DRF6*((E6-I)/CT)
IF(I.LT.D6)DRF6=0
URF7=COS(A*((U-T7)**2))
IF(I.LT.D7+CT)URF7=URF7*((I-D7)/CT)
IF(I.GT.E7-CT)URF7=URF7*((E7-I)/CT)
IF(I.LT.D7)URF7=0
DRF7=COS(A*((D+T7)**2))
IF(I.LT.D7+CT)DRF7=DRF7*((I-D7)/CT)
IF(I.GT.E7-CT)DRF7=DRF7*((E7-I)/CT)
IF(I.LT.D7)DRF7=0
URF8=COS(A*((U-T8)**2))
IF(I.LT.D8+CT)URF8=URF8*((I-D8)/CT)
IF(I.GT.E8-CT)URF8=URF8*((E8-I)/CT)
IF(I.LT.D8)URF8=0
DRF8=COS(A*((D+T8)**2))
IF(I.LT.D8+CT)DRF8=DRF8*((I-D8)/CT)
IF(I.GT.E8-CT)DRF8=DRF8*((E8-I)/CT)
IF(I.LT.D8)DRF8=0
URF9=COS(A*((U-T9)**2))
IF(I.LT.D9+CT)URF9=URF9*((I-D9)/CT)
IF(I.GT.E9-CT)URF9=URF9*((E9-I)/CT)

```

```

IF(I.LT.D9)URF9=0
DRF9=COS(A*((D+T9)**2))
IF(I.LT.D9+CT)DRF9=DRF9*((I-D9)/CT)
IF(I.GT.E9-CT)DRF9=DRF9*((E9-I)/CT)
IF(I.LT.D9)DRF9=0
URF10=COS(A*((U-T10)**2))
IF(I.LT.D10+CT)URF10=URF10*((I-D10)/CT)
IF(I.GT.E10-CT)URF10=URF10*((E10-I)/CT)
IF(I.LT.D10)URF10=0
DRF10=COS(A*((D+T10)**2))
IF(I.LT.D10+CT)DRF10=DRF10*((I-D10)/CT)
IF(I.GT.E10-CT)DRF10=DRF10*((E10-I)/CT)
IF(I.LT.D10)DRF10=0
X=UP1*(JRF1+URF2+URF3+URF4+URF5+URF6+URF7+URF8+URF9+URF10)
Y=X+FC*Y
R=DN1*(DRF1+DRF2+DRF3+DRF4+DRF5+DRF6+DRF7+DRF8+DRF9+DRF10)
S=R+FC*S
K=K+1
IF(K.EQ.SR)UP(J)=Y
IF(K.EQ.SR)DN(J)=S
IF(K.EQ.SR)J=J+1
IF(K.EQ.SR)K=0
20 CONTINUE

```

C Section 7

C In this section the order of the down sweep is reversed.

```

CENTER=(START+END)/2
DO 21 I=START,CENTER
TEMP=DN(I)
DN(I)=DN(END+START-I)
DN(END+START-I)=TEMP
21 CONTINUE

```

C Section 8

C In this section the up sweep and the down sweep are written
C to FOR08.DAT and FOR09.DAT respectively.

```

WRITE(8,23)(UP(J),J=0,END)
WRITE(9,23)(DN(J),J=0,END)
23 FORMAT(10F13.7)

```

```

STOP
END

```

C -----VIBRD1.FOR-----

C George Handley May 1978

C This program inverse fast Fourier transforms the up sweep
C written to FOR08.DAT from the program VIB10.FOR.

C All input parameters needed by the program are read from
C FOR07.DAT. The IMSL library programs FFT2 and FFRDR2 are
C used to execute the transformation. There is one output
C to this program, the real part of the transformed array.
C This is written to FOR14.DAT. A plot of the up sweep and
C the real part together can be had by executing PLOTV1.FOR
C after this program. This program is divided into five sections.
C A comment before each section briefly identifies it.
C For a more complete analysis, refer
C to the users manual for thesis # T-2082, on file with
C the Geophysics Department, Colorado School of Mines.

```

      COMPLEX SWEEP(8192),GAMN
      DIMENSION IWK(8193),UP(1876)
      INTEGER END

```

C Section 1

C In this section all parameters needed by the program
C will be read from FOR07.DAT and the up sweep will be read
C from FOR08.DAT.

```

1      FORMAT(G)
      READ(7,1)HI
      READ(7,1)END
      READ(7,1)M
      M=M+1
      READ(7,1)START
      READ(7,1)PI
      READ(7,1)ALPHA
      NDATA=2**M
      NALIAS=NDATA/2
      START=START+1
      END=END+1
      READ(8,2)(UP(J),J=1,END)
2      FORMAT(10F13.7)

```

C Section 2

C In this section the up sweep is multiplied with a Hanning
C truncator.

```

      X=(START+END)/2
      Y=PI/(END-START)
      DO 4 I=START,END
4      UP(I)=UP(I)*COS((I-X)*Y)

```

C Section 3

C In this section UP is loaded into the first half of the
 C complex array SWEEP, and then UP is folded into the
 C second half of SWEEP.

```

    DO 20 N=1,END
20   SWEEP(N)=CMPLX(UP(N),0.)
    DO 21 N=END+1,NALIAS
21   SWEEP(N)=CMPLX(0.,0.)
    NFOLD=NALIAS+1
    K=0
    SWEEP(NFOLD)=CMPLX(0.,0.)
    DO 22 N=NFOLD+1,NDATA
    K=K+1
22   SWEEP(N)=SWEEP(NFOLD-K)
  
```

C Section 4

C In this section the complex array SWEEP is fast Fourier
 C transformed using the IMSL library programs FFT2 and FFRDR2.

```

    DO 24 N=1,NDATA
24   SWEEP(N)=CONJG(SWEEP(N))
    CALL FFT2(SWEEP,M,IWK)
    CALL FFRDR2(SWEEP,M,IWK)
    DO 26 N=1,NDATA
26   SWEEP(N)=CONJG(SWEEP(N))
    DATA=NDATA
    DO 28 N=1,NDATA
28   SWEEP(N)=SWEEP(N)/CMPLX(DATA,0.)
  
```

C Section 5

C The real part of the transformed array is now written
 C to FOR14.DAT.

```

    DO 30 I=1,1876
30   UP(I)=REAL(SWEEP(I))
    WRITE(14,2)(UP(I),I=1,1876)

    STOP
    END
  
```

C -----PLOTV1.FOR-----

C George Handley May 1978

C This program plots the filtered decimated product of the up
C sweep from VIB10.FOR and the real part of the Fourier transform
C from VIBR01.FOR.

C One question is asked by this program to determine the
C amount of data (seconds) you want on the transformed axis.
C Everything else needed by the program is read from
C data files. The parameters are read from FOR07.DAT, the
C up sweep is read from FOR08.DAT, and the real part of the
C transformed array is read from FOR14.DAT. This program is
C divided into five sections. A comment before each
C section briefly identifies it. For a more complete
C description refer to the users manual for thesis # T-2082,
C on file with the Geophysics Department, Colorado School
C of Mines.

```

DIMENSION UP(2000),RE(2000),ITITLE(5),ITIT(4)
INTEGER END
DATA ITITLE/'INSTA','VTANE','OUS F','REQUE','NCY'/
DATA ITIT/'RECOR','D TIM','E (SE','C)'/

```

C Section 1

C In this section a question of how much time you want
C plotted on the transformed axis is asked. All constants
C needed by the program are read from FOR07.DAT. The up sweep
C is read from FOR08.DAT and the real part of
C the transformed array (RE) is read from FOR14.DAT.

```

WRITE(4,5)
5   FORMAT(' SECONDS OF DATA ON PLOT =',5)
   READ(4,1)SEC
1   FORMAT(G)
   READ(7,1)HI
   READ(7,1)END
   READ(7,1)M
   M=M+1
   NDATA=2**M
   REPTIM=END/HI
   REPFRE=NDATA/REPTIM
   NDATA=SEC*REPFRE
   READ(8,2)(UP(I),I=1,END+1)
   READ(14,2)(RE(I),I=1,NDATA)
2   FORMAT(10F13.7)

```

C Section 2

C In this section the UP array is tested and normalized.

```

      A=0
      DO 10 I=1,END+1,2
      B=AMAX1(ABS(UP(I)),ABS(UP(I+1)))
      IF(B.GT.A)A=B
10      CONTINUE
      DO 15 I=1,END+1
15      UP(I)=UP(I)/A

```

C Section 3

C In this section the RE array is tested and normalized.

```

      A=0
      DO 20 I=1,END+1,2
      B=AMAX1(ABS(RE(I)),ABS(RE(I+1)))
      IF(B.GT.A)A=B
20      CONTINUE
      DO 25 I=1,END+1
25      RE(I)=RE(I)/A

```

C Section 4

C In this section the UP array is plotted with axis and title.

```

      IF(IPLOT(10).NE.0)STOP"OPPS"
      J=NEWPEN(2)
      CALL AXES(0.,4.5,ITITLE,23,9.,0.0,0.,HI/9.,
* 9/HI,0,-1,5)
      CALL PLOT(0.,1.,3)
      CALL PLOT(0.,8.,2)
      J=NEWPEN(1)
      CALL PLOT(0.,6.,-3)
      DO 100 N=1,END+1
      X=(N-1)*9./(END+1)
      Y=UP(N)
      IF(N.EQ.1)CALL PLOT(X,Y,3)
      IF(N.EQ.2)CALL PLOT(X,Y,2)
      IF(N.GT.2)CALL PLOT(X,Y,0)
100  CONTINUE

```

C Section 5

C In this section the RE array is plotted with axis and title.

```

      J=NEWPEN(2)
      CALL AXES(0.,-5.,ITIT,17,9.,0.0,0.,SEC/9.,
* .9/SEC,0,1,1)
      J=NEWPEN(1)
      CALL PLOT(0.,-3.5,-3)
      DO 101 N=1,NDATA
      X=(N-1)*9./NDATA
      Y=RE(N)
      IF(N.EQ.1)CALL PLOT(X,Y,3)
      IF(N.EQ.2)CALL PLOT(X,Y,2)

```

```
101  IF(N.GT.2)CALL PLOT(X,Y,0)
      CONTINUE

      CALL PLOT(X,Y,999)
      STOP
      END
```

C -----VIBRO2.FOR-----

C George Handley May 1978

C This program inverse fast Fourier transforms the up sweep
C with the down sweep from VIB10.FOR. There are four outputs,
C the real and imaginary parts of the transformed array,
C and the real and imaginary parts after being multiplied
C by weighting factors.

C All input parameters needed by the program are read from
C FOR07.DAT. The filtered sampled up and down sweeps
C are read from FOR08.DAT and FOR09.DAT respectively. The
C IMSL library programs FFT2 and FFRDR2 are used to execute
C the transformation. The real and imaginary parts of the
C transformed array are written to FOR14.DAT and FOR15.DAT
C respectively. After being multiplied by a weighting
C function, the real and imaginary parts are written to
C FOR16.DAT and FOR17.DAT respectively. A plot of the
C sweeps and the transformed arrays can be had by executing
C PLOTV2.FOR after this program. A plot of the weighted
C transformed arrays and the sum of the two can be had by
C executing PLOTV3.FOR after this program. This
C program is divided into seven sections. A comment before
C each section briefly identifies it. For a more complete
C description refer to the users manual for thesis # T-2082,
C on file with the Geophysics Department, Colorado School
C of Mines.

```

COMPLEX SWEEP(8192),GAMN
DIMENSION IWK(8193),UP(1876),DN(1876)
INTEGER END

```

C Section 1

C In this section the program reads constants
C written to FOR07.DAT from VIB10.FOR. These are needed
C to create new constants and to format the data for
C execution of the Fourier transform.

```

1  FORMAT(G)
   READ(7,1)HI
   READ(7,1)END
   READ(7,1)M
   M=M+1
   READ(7,1)START
   READ(7,1)PI
   READ(7,1)ALPHA
   READ(7,1)FILTCO
   NDATA=2**M
   NALIAS=NDATA/2
   REPTIM=END/HI

```

```

REPFRE=HI/END*NDATA
A1=ALPHA/(REPFRE**2)
TANG=2*ALPHA*REPTIM/(NDATA*FILTCO)
W=(2*ALPHA*REPTIM/NDATA)**2
F=FILTCO**2

```

C Section 2

C In this section the up sweep and the down sweep are read
C from FOR08.DAT and FOR09.DAT.

```

START=START+1
END=END+1
READ(8,2)(UP(J),J=1,END)
READ(9,2)(DN(J),J=1,END)
2  FORMAT(10F13.7)

```

C Section 3

C In this section both arrays are multiplied by
C a Hanning truncator.

```

X=(START+END)/2
Y=PI/(END-START)
DO 4 I=START,END
UP(I)=UP(I)*COS((I-X)*Y)
DN(I)=DN(I)*COS((I-X)*Y)
4  CONTINUE

```

C Section 4

C In this section the up sweep is loaded into the real part
C of the first half of the complex array "SWEEP" and the down
C sweep is loaded into the real part of the folded half. The
C imaginary half is zero.

```

DO 20 N=1,END
20  SWEEP(N)=CMPLX(UP(N),0.)
DO 21 N=END+1,NALIAS
21  SWEEP(N)=CMPLX(0.,0.)
NFOLD=NALIAS+1
SWEEP(NFOLD)=CMPLX(0.,0.)
K=0
DO 22 N=1,END
SWEEP(NDATA-K)=CMPLX(DN(N),0.)
K=K+1
22  CONTINUE
DO 23 N=END+1,NALIAS-1
23  SWEEP(NDATA-N)=CMPLX(0.,0.)

```

C Section 5

C In this section the data is fast Fourier transformed using
C FFT2 and FFRDR2. (IMSL programs).

```

      DO 24 N=1,NDATA
24     SWEEP(N)=CONJG(SWEEP(N))
      CALL FFT2(SWEEP,M,IWK)
      CALL FFRDR2(SWEEP,M,IWK)
      DO 26 N=1,NDATA
26     SWEEP(N)=CONJG(SWEEP(N))
      DATA=NDATA
      DO 28 N=1,NDATA
28     SWEEP(N)=SWEEP(N)/CMPLX(DATA,0.)

```

C Section 6

C In this section the real and imaginary parts of the transformed
C array are written to FOR14.DAT and FOR15.DAT respectively.

```

      DO 30 I=1,1876
      UP(I)=REAL(SWEEP(I))
      DN(I)=AIMAG(SWEEP(I))
30     CONTINUE
      WRITE(14,2)(UP(I),I=1,1876)
      WRITE(15,2)(DN(I),I=1,1876)

```

C Section 7

C In this section the real and imaginary parts are multiplied by
C their weighting factors and the products written to FOR16.DAT
C and FOR17.DAT respectively.

```

      DO 32 I=1,1876
      PHI=ATAN(TANG*(I-1))
      AMP=(F+W*(I-1)**2)**.5
      UP(I)=UP(I)*AMP*COS(A1*((I-1)**2)+PHI)
      DN(I)=DN(I)*AMP*SIN(A1*((I-1)**2)+PHI)
32     CONTINUE
      WRITE(16,2)(UP(I),I=1,1876)
      WRITE(17,2)(DN(I),I=1,1876)

```

```

STOP
END

```

C -----PLOTV2.FOR-----

C George Handley May 1978

C This program plots the products of the up sweep and the
C down sweep from VIB10.FOR or NOFL10.FOR. It also plots the
C real and imaginary parts of the transformed arrays from
C VIBR02.FOR or NOFILT.FOR.

C One question is asked by this program to determine the
C amount of data (seconds) you want on the transformed axis.
C Everything else needed by the program is read from data
C files. The parameters are read from FOR07.DAT, the up sweep
C from FOR08.DAT, the down sweep from FOR08.DAT, the real
C part of the transformed array from FOR14.DAT, and the
C imaginary part from FOR15.DAT. The output is plotted on the
C Houston DP-12 plotter. This program is divided into six sections.
C A comment before each section briefly identifies it. For a
C more complete description refer to the users manual for
C thesis # T-2082, on file with the Geophysics Department,
C Colorado School of Mines.

```

DIMENSION RE(2001),AI(2001),UP(2001),DN(2001),IT(4),ITI(5)
INTEGER END
DATA IT/'RECOR','D TIM','E (SE','C)'/
DATA ITI/'INSTA','NTANE','OUS F','REQUE','NCY'/

```

C Section 1

C In this section a question of how much time you want plotted
C on the transformed axis is asked. All constants needed by the
C program are read from FOR07.DAT. The up sweep and down sweep
C are read from FOR08.DAT and FOR09.DAT respectively. The real (RE)
C and imaginary (AI) parts of the transformed array are read from
C FOR14.DAT and FOR15.DAT respectively.

```

WRITE(4,4)
4  FORMAT(' SECONDS OF DATA ON PLOT=',I)
   READ(4,1)SEC
   READ(7,1)HI
   READ(7,1)END
   READ(7,1)M
   M=M+1
1  FORMAT(3)
   NDATA=2**M
   REPFRE=HI/END*NDATA
   REPTIM=END/HI
   NDATA=SEC*REPFRE
   READ(8,2)(UP(I),I=1,END+1)
   READ(9,2)(DN(I),I=1,END+1)
   READ(14,2)(RE(I),I=1,NDATA)
   READ(15,2)(AI(I),I=1,NDATA)

```

```
2      FORMAT(10F13.7)
```

```
C Section 2
```

```
C In this section the UP array and the DN array are normalized.
```

```
      A=0
      B=0
      DO 10 I=1,END+1,2
      C=AMAX1(ABS(UP(I)),ABS(UP(I+1)))
      D=AMAX1(ABS(DN(I)),ABS(DN(I+1)))
      IF(C.GT.A)A=C
      IF(D.GT.B)B=D
10     CONTINUE
      DO 15 I=1,END+1
      UP(I)=UP(I)/A
      DN(I)=DN(I)/B
15     CONTINUE
```

```
C Section 3
```

```
C In this section the RE array and the AI array are normalized.
```

```
      A=0
      B=0
      DO 20 I=1,NDATA,2
      C=AMAX1(ABS(RE(I)),ABS(RE(I+1)))
      D=AMAX1(ABS(AI(I)),ABS(AI(I+1)))
      IF(C.GT.A)A=C
      IF(D.GT.B)B=D
20     CONTINUE
      DO 25 I=1,NDATA
      RE(I)=RE(I)/A
      AI(I)=AI(I)/B
25     CONTINUE
```

```
C Section 4
```

```
C In this section the axis and titles for each array are plotted.
```

```
      IF(IPLOT(10).NE.0.)STOP'DPPS'
      J=NEWPEN(2)
      CALL AXES(8.75,1.,ITI,17,7.,90.0,0.,SEC/7.,
* 7/SEC,0,1,1)
      CALL AXES(5.,1.,ITI,23,7.,90.0,0.,HI/7.,
* 7/HI,0,-1,5)
      CALL AXES(2.5,1.,ITI,23,7.,90.0,-50.,HI/7.,
* 7/HI,0,-1,5)
      CALL PLOT(0.,1.,3)
      CALL PLOT(8.75,1.,2)
```

C Section 5

C In this section the UP array and the DN array are plotted.

```

      J=NEWPEN(1)
      CALL PLOT(1.25,0.,-3)
      DO 100 N=1,END+1
      Y=(N-1)*7./END+1
      X=-DN(END+2-N)
      IF(N.EQ.1)CALL PLOT(X,Y,3)
      IF(N.EQ.2)CALL PLOT(X,Y,2)
      IF(N.GT.2)CALL PLOT(X,Y,0)
100   CONTINUE
      CALL PLOT(2.5,0.,-3)
      DO 101 N=1,END+1
      Y=(N-1)*7./END+1
      X=-UP(N)
      IF(N.EQ.1)CALL PLOT(X,Y,3)
      IF(N.EQ.2)CALL PLOT(X,Y,2)
      IF(N.GT.2)CALL PLOT(X,Y,0)
101   CONTINUE

```

C Section 6

C In this section the RE array and the AI array are plotted.

```

      CALL PLOT(2.25,1.,-3)
      DO 102 N=1,NDATA
      Y=(N-1)*7./NDATA
      X=-RE(N)
      IF(N.EQ.1)CALL PLOT(X,Y,3)
      IF(N.EQ.2)CALL PLOT(X,Y,2)
      IF(N.GT.2)CALL PLOT(X,Y,0)
102   CONTINUE
      CALL PLOT(1.75,0.,-3)
      DO 103 N=1,NDATA
      Y=(N-1)*7./NDATA
      X=AI(N)
      IF(N.EQ.1)CALL PLOT(X,Y,3)
      IF(N.EQ.2)CALL PLOT(X,Y,2)
      IF(N.GT.2)CALL PLOT(X,Y,0)
103   CONTINUE
      CALL PLOT(X,Y,999)

      STOP
      END

```

C -----PLDTV3.FOR-----

C George Handley May 1978

C This program plots the real and imaginary parts of the
C transformed array from VIBRQ2.FOR or NOFILT.FOR after they
C have been multiplied by their respective weighting functions.
C It also plots the sum of these two parts.

C One question is asked by this program to determine the
C amount of data (seconds) you want on the transformed axis.
C Everything else needed by the program is read from data
C files. The parameters are read from FOR07.DAT, and the
C weighted real and imaginary parts are read from FOR16.DAT
C and FOR17.DAT respectively. This program is divided into
C six sections. A comment before each section briefly identifies
C it. For a more complete description refer to the users
C manual for thesis # T-2082, on file with the Geophysics
C Department, Colorado School of Mines.

```

      DIMENSION RE(2001),AI(2001),ITITLE(4),SUM(2001)
      DATA ITITLE/'RECOR','D TIM','E (SE','C)'/

```

C Section 1

C In this section a question of how much time you want plotted
C on the transformed axis is asked. All constants needed
C by the program are read from FOR07.DAT or are computed.
C The amplitude and phase corrected real (RE) and imaginary (AI) parts
C of the transformed array are read from FOR16.DAT and FOR17.DAT
C respectively.

```

      WRITE(4,4)
4      FORMAT(' SECONDS OF DATA ON PLOT=',4)
      READ(4,5)SEC
5      FORMAT(F)
      READ(7,1)HI
      READ(7,1)END
      READ(7,1)M
      M=M+1
1      FORMAT(G)
      NDATA=2**M
      REPFRE=HI/END*NDATA
      REPTIM=END/HI
      NDATA=SEC*REPFRE
      READ(16,2)(RE(I),I=1,NDATA)
      READ(17,2)(AI(I),I=1,NDATA)
2      FORMAT(10F13.7)

```

C Section 2

C In this section the RE and AI arrays are summed and
C written to the array SUM.

```

      DO 3 I=1,NDATA
3      SUM(I)=RE(I)-AI(I)

```

C Section 3

C In this section the RE, AI, and SUM arrays are normalized.

```

      A=0
      B=0
      C=0
      DO 10 I=1,NDATA,2
      D=AMAX1(ABS(RE(I)),ABS(RE(I+1)))
      E=AMAX1(ABS(AI(I)),ABS(AI(I+1)))
      F=AMAX1(ABS(SUM(I)),ABS(SUM(I+1)))
      IF(D.GT.A)A=D
      IF(E.GT.B)B=E
      IF(F.GT.C)C=F
10     CONTINUE
      DO 15 I=1,NDATA
      RE(I)=RE(I)/A
      AI(I)=AI(I)/B
      SUM(I)=SUM(I)/C
15     CONTINUE

```

C Section 4

C In this section the axis, with title is plotted.

```

      IF(IPLOT(10).NE.0.)STOP"OPPS"
      J=NEWPEN(2)
      CALL AXES(8.75,1.,ITITLE,17,7.,90.0,0.,SEC/7.,
* 7/SEC,0,1,1)
      CALL PLOT(0.,1.,3)
      CALL PLJT(8.75,1.,2)

```

C Section 5

C In this section the normalized RE and AI arrays are plotted.

```

      J=NEWPEN(1)
      CALL PLOT(2.,1.,-3)
      DO 100 N=1,NDATA
      Y=(N-1.)*7./NDATA
      X=-RE(N)
      IF(N.EQ.1)CALL PLOT(X,Y,3)
      IF(N.EQ.2)CALL PLOT(X,Y,2)
      IF(N.GT.2)CALL PLOT(X,Y,0)
100    CONTINUE
      CALL PLOT(2.,0.,-3)
      DO 101 N=1,NDATA
      Y=(N-1.)*7./NDATA
      X=AI(N)
      IF(N.EQ.1)CALL PLOT(X,Y,3)

```

```
        IF(N.EQ.2)CALL PLOT(X,Y,2)  
        IF(N.GT.2)CALL PLOT(X,Y,0)  
101    CONTINUE
```

C Section 6

C In this section the normalized SUM array is plotted.

```
        CALL PLOT(2.75,0.,-3)  
        DO 102 V=1,NDATA  
        Y=(N-1)*7./NDATA  
        X=-2*SUM(N)  
        IF(N.EQ.1)CALL PLOT(X,Y,3)  
        IF(N.EQ.2)CALL PLOT(X,Y,2)  
        IF(N.GT.2)CALL PLOT(X,Y,0)  
102    CONTINUE  
        CALL PLOT(X,Y,999)  
  
        STOP  
        END
```

C -----NOFL10.FOR-----

C George Handley May 1978

C This program generates a non-filtered sampled product
C that is analogous to the VIBROLOKATOR II method of
C recording field data. The output to this can be input
C into NOFILT.FOR to inverse fast Fourier transform
C the data.

C Sixteen questions are asked by this program. These questions
C either define or are used to define all constants and
C variables needed by the program. The answers to these
C questions may be typed in integer or floating point format.
C The output off this program is written to three data
C files for use in future programs. All constants needed
C to define future constants and variables are written to
C FOR07.DAT. The sampled up sweep and down sweep are written
C to FOR08.DAT and FOR09.DAT respectively. This program is
C divided into eight sections. A comment before each
C section briefly defines it. For a more complete description
C refer to the users manual for thesis # T-2082, on file
C with the Geophysics Department, Colorado School of Mines.

```

DIMENSION UP(0/2000),JN(0/2000)
REAL LOW
INTEGER END
PI=3.141592654

```

C Section 1

C In this section all questions are asked to determine the
C constants needed by the program.

```

WRITE(4,1)
1  FORMAT(" INITIAL FREQUENCY=", $)
   READ(4,2)LOW
2  FORMAT(F)
   WRITE(4,3)
3  FORMAT(" FINAL FREQUENCY=", $)
   READ(4,2)HI
   WRITE(4,4)
4  FORMAT(" TIME=", $)
   READ(4,2)TIME
   WRITE(4,5)
5  FORMAT(" SAMPLES PER SECOND=", $)
   READ(4,2)SPS
   WRITE(4,6)
6  FORMAT(" FIRST REFLECTION TIME=", $)
   READ(4,2)T1
   WRITE(4,7)
7  FORMAT(" SECOND REFLECTION TIME=", $)

```

```

      READ(4,2)T2
      WRITE(4,8)
8     FORMAT(" THIRD REFLECTION TIME=", $)
      READ(4,2)T3
      WRITE(4,40)
40    FORMAT(" FOURTH REFLECTION TIME=", $)
      READ(4,2)T4
      WRITE(4,41)
41    FORMAT(" FIFTH REFLECTION TIME=", $)
      READ(4,2)T5
      WRITE(4,42)
42    FORMAT(" SIXTH REFLECTION TIME=", $)
      READ(4,2)T6
      WRITE(4,43)
43    FORMAT(" SEVENTH REFLECTION TIME=", $)
      READ(4,2)T7
      WRITE(4,44)
44    FORMAT(" EIGHTH REFLECTION TIME=", $)
      READ(4,2)T8
      WRITE(4,45)
45    FORMAT(" NINETH REFLECTION TIME=", $)
      READ(4,2)T9
      WRITE(4,46)
46    FORMAT(" TENTH REFLECTION TIME=", $)
      READ(4,2)T10
      WRITE(4,9)
9     FORMAT(" COUPLING TIME=", $)
      READ(4,2)CT
      WRITE(4,11)
11    FORMAT(" SAMPLE RATE=", $)
      READ(4,2)SR

```

C Section 2

C In this section additional constants will be computed
 C from the answers given already, and those needed for future
 C programs will be written to FOR07.DAT.

```

      LAST=SPS*TIME
      GAP=LOW*LAST/(HI-LOW)
      CT=CT*SPS
      TOTAL=GAP+LAST
      TOTIME=TIME+GAP/SPS
      END=TOTAL/SR
      START=GAP/SR
      ALPHA=PI*HI/TOTIME
      A=ALPHA/(SPS**2)
      M=1
12    IF(2**M.GT.END)GO TO 13
      M=M+1
      GO TO 12
13    M=M+1

```

```

14  FORMAT(G)
    WRITE(7,14)HI
    WRITE(7,14)END
    WRITE(7,14)M
    WRITE(7,14)START
    WRITE(7,14)PI
    WRITE(7,14)ALPHA

```

C Section 3

C In this section the times governing the parameters of each
C sweep will be defined or redefined in terms of samples
C per second.

```

T1=T1*SPS
T2=T2*SPS
T3=T3*SPS
T4=T4*SPS
T5=T5*SPS
T6=T6*SPS
T7=T7*SPS
T8=T8*SPS
T9=T9*SPS
T10=T10*SPS
  D1=GAP+T1
D2=GAP+T2
D3=GAP+T3
D4=GAP+T4
D5=GAP+T5
D6=GAP+T6
D7=GAP+T7
D8=GAP+T8
D9=GAP+T9
D10=GAP+T10
E1=TOTAL+T1
E2=TOTAL+T2
E3=TOTAL+T3
E4=TOTAL+T4
E5=TOTAL+T5
E6=TOTAL+T6
E7=TOTAL+T7
E8=TOTAL+T8
E9=TOTAL+T9
E10=TOTAL+T10

```

C Section 4

C This do loop assigns a value of zero to all the points that
C would correspond to the truncated section of the sweep from
C zero Hz to LOW.

```

DO 19 I=0,START-1
  UP(I)=0

```

```

      DN(I)=0
19  CONTINUE

```

C Section 5

C These next five lines define counters needed by the program.

```

      H=TOTAL+GAP
      J=START
      K=SR-1
      Y=0
      S=0

```

C Section 6

C This do loop generates, sums, multiplies, and

C samples the data. Both for the up sweep and the down sweep.

```

      DO 20 I=GAP,TOTAL
      U=I
      D=H-I
      UP1=COS(A*(U**2))
      IF(I.LT.GAP+CT)UP1=UP1*((I-GAP)/CT)
      IF(I.GT.TOTAL-CT)UP1=UP1*((TOTAL-I)/CT)
      DN1=COS(A*(D**2))
      IF(I.LT.GAP+CT)DOWN=DOWN*((I-GAP)/CT)
      IF(I.GT.TOTAL-CT)DOWN=DOWN*((TOTAL-I)/CT)
      URF1=COS(A*((U-T1)**2))
      IF(I.LT.D1+CT)URF1=URF1*((I-D1)/CT)
      IF(I.GT.E1-CT)URF1=URF1*((E1-I)/CT)
      IF(I.LT.D1)URF1=0
      DRF1=COS(A*((D+T1)**2))
      IF(I.LT.D1+CT)DRF1=DRF1*((I-D1)/CT)
      IF(I.GT.E1-CT)DRF1=DRF1*((E1-I)/CT)
      IF(I.LT.D1)DRF1=0
      URF2=COS(A*((U-T2)**2))
      IF(I.LT.D2+CT)URF2=URF2*((I-D2)/CT)
      IF(I.GT.E2-CT)URF2=URF2*((E2-I)/CT)
      IF(I.LT.D2)URF2=0
      DRF2=COS(A*((D+T2)**2))
      IF(I.LT.D2+CT)DRF2=DRF2*((I-D2)/CT)
      IF(I.GT.E2-CT)DRF2=DRF2*((E2-I)/CT)
      IF(I.LT.D2)DRF2=0
      URF3=COS(A*((U-T3)**2))
      IF(I.LT.D3+CT)URF3=URF3*((I-D3)/CT)
      IF(I.GT.E3-CT)URF3=URF3*((E3-I)/CT)
      IF(I.LT.D3)URF3=0
      DRF3=COS(A*((D+T3)**2))
      IF(I.LT.D3+CT)DRF3=DRF3*((I-D3)/CT)
      IF(I.GT.E3-CT)DRF3=DRF3*((E3-I)/CT)
      IF(I.LT.D3)DRF3=0
      URF4=COS(A*((U-T4)**2))
      IF(I.LT.D4+CT)URF4=URF4*((I-D4)/CT)

```

```

IF(I.GT.E4-CT)URF4=URF4*((E4-I)/CT)
IF(I.LT.D4)URF4=0
DRF4=COS(A*((D+T4)**2))
IF(I.LT.D4+CT)DRF4=DRF4*((I-D4)/CT)
IF(I.GT.E4-CT)DRF4=DRF4*((E4-I)/CT)
IF(I.LT.D4)DRF4=0
URF5=COS(A*((U-T5)**2))
IF(I.LT.D5+CT)URF5=URF5*((I-D5)/CT)
IF(I.GT.E5-CT)URF5=URF5*((E5-I)/CT)
IF(I.LT.D5)URF5=0
DRF5=COS(A*((D+T5)**2))
IF(I.LT.D5+CT)DRF5=DRF5*((I-D5)/CT)
IF(I.GT.E5-CT)DRF5=DRF5*((E5-I)/CT)
IF(I.LT.D5)DRF5=0
URF6=COS(A*((U-T6)**2))
IF(I.LT.D6+CT)URF6=URF6*((I-D6)/CT)
IF(I.GT.E6-CT)URF6=URF6*((E6-I)/CT)
IF(I.LT.D6)URF6=0
DRF6=COS(A*((D+T6)**2))
IF(I.LT.D6+CT)DRF6=DRF6*((I-D6)/CT)
IF(I.GT.E6-CT)DRF6=DRF6*((E6-I)/CT)
IF(I.LT.D6)DRF6=0
URF7=COS(A*((U-T7)**2))
IF(I.LT.D7+CT)URF7=URF7*((I-D7)/CT)
IF(I.GT.E7-CT)URF7=URF7*((E7-I)/CT)
IF(I.LT.D7)URF7=0
DRF7=COS(A*((D+T7)**2))
IF(I.LT.D7+CT)DRF7=DRF7*((I-D7)/CT)
IF(I.GT.E7-CT)DRF7=DRF7*((E7-I)/CT)
IF(I.LT.D7)DRF7=0
URF8=COS(A*((U-T8)**2))
IF(I.LT.D8+CT)URF8=URF8*((I-D8)/CT)
IF(I.GT.E8-CT)URF8=URF8*((E8-I)/CT)
IF(I.LT.D8)URF8=0
DRF8=COS(A*((D+T8)**2))
IF(I.LT.D8+CT)DRF8=DRF8*((I-D8)/CT)
IF(I.GT.E8-CT)DRF8=DRF8*((E8-I)/CT)
IF(I.LT.D8)DRF8=0
URF9=COS(A*((U-T9)**2))
IF(I.LT.D9+CT)URF9=URF9*((I-D9)/CT)
IF(I.GT.E9-CT)URF9=URF9*((E9-I)/CT)
IF(I.LT.D9)URF9=0
DRF9=COS(A*((D+T9)**2))
IF(I.LT.D9+CT)DRF9=DRF9*((I-D9)/CT)
IF(I.GT.E9-CT)DRF9=DRF9*((E9-I)/CT)
IF(I.LT.D9)DRF9=0
URF10=COS(A*((U-T10)**2))
IF(I.LT.D10+CT)URF10=URF10*((I-D10)/CT)
IF(I.GT.E10-CT)URF10=URF10*((E10-I)/CT)
IF(I.LT.D10)URF10=0
DRF10=COS(A*((D+T10)**2))

```

```

IF(I.LT.D10+CT)DRF10=DRF10*((I-D10)/CT)
IF(I.GT.E10-CT)DRF10=DRF10*((E10-I)/CT)
IF(I.LT.D10)DRF10=0
X=UP1*(URF1+URF2+URF3+URF4+URF5+URF6+URF7+URF8+URF9+URF10)
R=DN1*(DRF1+DRF2+DRF3+DRF4+DRF5+DRF6+DRF7+DRF8+DRF9+DRF10)
K=K+1
IF(K.EQ.SR)UP(J)=X
IF(K.EQ.SR)DN(J)=R
IF(K.EQ.SR)J=J+1
IF(K.EQ.SR)K=0
20 CONTINUE

```

C Section 7

C In this section the order of the down sweep is reversed.

```

CENTER=(START+END)/2
DO 21 I=START,CENTER
TEMP=DN(I)
DN(I)=DN(END+START-I)
DN(END+START-I)=TEMP
21 CONTINUE

```

C Section 8

C In this section the up sweep and the down sweep are written
C to FOR08.DAT and FOR09.DAT respectively.

```

WRITE(8,23)(UP(J),J=0,END)
WRITE(9,23)(DN(J),J=0,END)
23 FORMAT(10F13.7)

```

```

STOP
END

```

C -----NOFILT.FOR-----

C George Handley May 1978

C This program inverse fast Fourier transforms the
 C non-filtered up and down sweeps from NOFL10.FOR. There
 C are four outputs, the real and imaginary parts to the
 C transformed array, and the real and imaginary parts after
 C their phase angles have been squared.

C All input parameters needed by the program are read from
 C FOR07.DAT. The non-filtered sampled up and down sweeps are
 C read from from FOR08.DAT and FOR09.DAT respectively. The
 C IMSL library programs FFT2 and FFRDR2 are used to execute
 C the transformation. The real and imaginary parts of the
 C transformed array are written to FOR14.DAT and FOR15.DAT
 C respectively. After being multiplied by their weighting
 C functions, the real and imaginary parts are written to
 C FOR16.DAT and FOR17.DAT respectively. A plot of the sweeps
 C and the transformed arrays can be had by executing PLOTV2.FOR
 C after this program. A plot of the transformed arrays after
 C squaring their phase angles, and a plot of the sum of the
 C two can be had by executing PLOTV3.FOR after this program.
 C This program is divided into seven sections. A comment
 C before each section briefly identifies it. For a more
 C complete description refer to the users manual for
 C thesis # T-2082, on file with the Geophysics Department,
 C Colorado School of Mines.

```

COMPLEX SWEEP(8192),GAMN
DIMENSION IWK(8193),UP(1876),DN(1876)
INTEGER END
  
```

C Section 1

C In this section the program reads constants written
 C to FOR07.DAT from NOFL10.FOR.
 C These are needed to create new constants and to format the
 C data for execution of the fast Fourier transform.

```

1   FORMAT(G)
    READ(7,1)HI
    READ(7,1)END
    READ(7,1)M
    M=M+1
    READ(7,1)START
    READ(7,1)PI
    READ(7,1)ALPHA
    NDATA=2**M
    NALIAS=NDATA/2
    REPTIM=END/HI
    REPFRE=HI/END*NDATA
  
```

```
A1=ALPHA/(REPFRE**2)
```

```
C Section 2
```

```
C In this section the non-filtered up sweep and down  
C sweep are read from FOR08.DAT and FOR09.DAT.
```

```
START=START+1  
END=END+1  
READ(8,2)(UP(J),J=1,END)  
READ(9,2)(DN(J),J=1,END)  
2   FORMAT(10F13.7)
```

```
C Section 3
```

```
C In this section both arrays are multiplied by a  
C Hanning truncator.
```

```
X=(START+END)/2  
Y=PI/(END-START)  
DO 4 I=START,END  
UP(I)=UP(I)*COS((I-X)*Y)  
DN(I)=DN(I)*COS((I-X)*Y)  
4   CONTINUE
```

```
C Section 4
```

```
C In this section the up sweep is loaded into the real part  
C of the first half of the complex array "SWEEP" and the  
C down sweep is loaded into the real part of the folded half.  
C The imaginary part is zero.
```

```
DO 20 N=1,END  
20  SWEEP(N)=CMPLX(UP(N),0.)  
DO 21 N=END+1,NALIAS  
21  SWEEP(N)=CMPLX(0.,0.)  
NFOLD=NALIAS+1  
SWEEP(NFOLD)=CMPLX(0.,0.)  
K=0  
DO 22 N=1,END  
SWEEP(N+DATA-K)=CMPLX(DN(N),0.)  
K=K+1  
22  CONTINUE  
DO 23 N=END+1,NALIAS-1  
23  SWEEP(N+DATA-N)=CMPLX(0.,0.)
```

```
C Section 5
```

```
C In this section the data is fast Fourier transformed  
C using FFT2 and FFRDR2 (IMSL programs).
```

```
DO 24 N=1,NDATA  
24  SWEEP(N)=CONJG(SWEEP(N))  
CALL FFT2(SWEEP,M,IWK)  
CALL FFRDR2(SWEEP,M,IWK)
```

```
      DO 26 N=1,NDATA
26     SWEEP(N)=CONJG(SWEEP(N))
      DATA=NDATA
      DO 28 N=1,NDATA
28     SWEEP(N)=SWEEP(N)/CMPLX(DATA,0.)
```

C Section 6

C In this section the real and imaginary parts of the transformed
C array are written to FOR14.DAT and FOR15.DAT respectively.

```
      DO 30 I=1,1876
      UP(I)=REAL(SWEEP(I))
      DN(I)=AIMAG(SWEEP(I))
30     CONTINUE
      WRITE(14,2)(UP(I),I=1,1876)
      WRITE(15,2)(DN(I),I=1,1876)
```

C Section 7

C In this section each array is multiplied by it's weighting
C function and the results are written to FOR16.DAT and
C FOR17.DAT.

```
      DO 32 I=1,1876
      UP(I)=UP(I)*COS(A1*((I-1)**2))
      DN(I)=DN(I)*SIN(A1*((I-1)**2))
32     CONTINUE
      WRITE(16,2)(UP(I),I=1,1876)
      WRITE(17,2)(DN(I),I=1,1876)
```

```
STOP
END
```

C -----ANALOG.FOR-----

C George E Handley May 1978

C This program is much like VIBRO1.FOR except that it
 C fast Fourier transforms the analog test data instead of
 C the computer generated sweeps. This data is written to
 C three data files, DATA9.DAT, DATA10.DAT, and DATA11.DAT.
 C To use this program you must first copy the file you want
 C transformed to DATA.DAT.

```

COMPLEX CDATA
DIMENSION CDATA (1024),IWK (1025),DATA(1024)
OPEN(UNIT=1,ACCESS="SEQIN",FILE="DATA.DAT")

```

C Section 1
 C In this section four questions will be asked to determine
 C constants needed to format the data.

```

WRITE(4,1)
1  FORMAT(" FREQUENCY INCREMENT=", $)
   READ(4,2)IFRE
2  FORMAT(I)
   WRITE(4,3)
3  FORMAT(" INITIAL FREQUENCY=", $)
   READ(4,2)IFRE1
   WRITE(4,4)
4  FORMAT(" FINAL FREQUENCY=", $)
   READ(4,2)IFREF
   WRITE(4,5)
5  FORMAT(" POWER OF 2=", $)
   READ(4,2)I
   IZERO=IFRE1/IFRE
   M=2**I
   L=(IFREF-IFRE1)/IFRE+1

```

C Section 2
 C In this section zeros will be inserted into the complex
 C array "CDATA" to augment the truncated section from
 C zero Hz to the initial frequency.

```

DO 10 N=1,IZERO
10  CDATA(N)=CMPLX(0.0,0.0)

```

C Section 3
 C In this section the data copied to DATA.DAT is loaded
 C into the complex array CDATA.

```

M7=IZERO+1
M1=M7+L

```

```

      READ(1,20)(CDATA(N1),N1=M7,M1)
20   FORMAT(5(F,F))
      CLOSE(UNIT=1)

```

C Section 4

C In this section the complex array CDATA is augmented
 C with zeros from the final frequency to the power of two.

```

      M2=M/2+1
      L1=M1+1
      DO 30 N2=L1,M2
30   CDATA(N2)=CMPLX(0.0,0.0)
      N7=2
      M3=M2+1

```

C Section 5

C In this section the fast Fourier transform is taken,
 C using the IMSL programs FFT2 and FFRDR2.

```

      DO 40 J=M3,M
      CDATA(J)=CONJG(CDATA(J-N7))
40   N7=N7+2
      CALL FFT2(CDATA,I,IWK)
      CONTINUE
      CALL FFRDR2(CDATA,I,IWK)
      CONTINUE
      TM=M
      DO 80 N5=1,M
      CDATA(N5)=CONJG(CDATA(N5))
      R=REAL(CDATA(N5))/TM
      AI=AIMAG(CDATA(N5))/TM
80   CDATA(N5)=CMPLX(R,AI)

```

C Section 6

C In this section both the real and the imaginary parts
 C of the transformed array are written to FOR02.DAT
 C complete with column headings and a time index to
 C identify your position in record time.

```

      WRITE(2,85)
85   FORMAT(7X,"TIME",14X,"REAL",10X,"IMAGINARY",18X,"REAL",
115X,"IMAGINARY"/)
      TIFRE=IFRE
      DELT=1/(TM*TIFRE)
      DO 100 N6=1,M,2
      TNG=N6-1
      TIME=TNG*DELT
      WRITE(2,90)TIME,CDATA(N6),CDATA(N6+1)
90   FORMAT(5X,F10.5,2(E20.5,E20.5,10X))
100  CONTINUE

```

C Section 7

C In this section the real array is tested and normalized.

```
      SCF=0.
      KC=0
      DO 150 J=1,M,2
      DATA(J)=REAL(CDATA(J))
      DATA(J+1)=REAL(CDATA(J+1))
      DT=AMAX1(ABS(DATA(J)),ABS(DATA(J+1)))
      IF(DT.GT.SCF)SCF=DT
150    CONTINUE
      DO 200 J=1,M
200    DATA(J)=DATA(J)/SCF
      IF(IPLOT(10).NE.0.)STOP 'OOPS'
```

C Section 8

C In this section the real array is plotted.

```
      CALL PLOT(0.,1.,3)
      CALL PLJT(7.,1.,2)
      CALL PLOT(3.,1.,-3)
      DO 201 N=1,M2
      Y=(N-1.)*9./M2
      X=DATA(N)
      IF(N.EQ.1)CALL PLOT(X,Y,3)
      IF(N.EQ.2)CALL PLOT(X,Y,2)
      IF(N.GT.2)CALL PLOT(X,Y,0)
201    CONTINUE
      CALL PLOT(X,Y,999)
      STOP
      END
```

References

- Brigham, E. Oran, 1974, The Fast Fourier Transform:
New Jersey, Prentice Hall, Inc.
- Claerbout, Jon F., 1976, Fundamentals of Geophysical
Data Processing: With Applications to Petroleum
Prospecting: New York, McGraw Hill, Inc., p. 1-23
- Geyer, R. L., 1970, Vibroseis Parameter Optimization: Oil
and Gas Journal, April.
- Harman, W. W., 1963, Principles of the Statistical Theory of
Communication: New York, McGraw Hill Book Company,
Inc., p. 156-166.
- International Mathematical and Statistics Library, 1977,
IMSL LIB 02-0006 VI: Houston, Texas, IMSL Subroutines
FFT2 and FFRDR2.
- Klauder, J. R., Price, A. C., Darlington, s., and
Albersheim, W. J., 1960, The Theory and Design of
Chirp Radars: The Bell System Technical Journal,
v. 39, no. 4, p. 745-808.
- McCracken, Daniel D., 1972, A Guide to Fortran IV
Programming: New York, John Wiley and Sons, Inc.
- Papoulis, Athanasios, 1962, The Fourier Integral and It's0
Applications: New York, McGraw Hill Book Company, Inc.
- Shanks, John L., 1967, Recursion Filters for Digital
Processing: Geophysics, v. 32, no. 1, p. 33-51.
- White, J. E., 1965, Seismic Waves: New York, McGraw Hill
Book Company, Inc.