T-2914

ITERATIVE STACK IN PETROLEUM SEISMOLOGY

by

CHIE-CHUNG YEN

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

T-2914

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

Golden, Colorado
Date Apr. 19, '84

Signed: Yen Chie-chung
Chie-chung Yen

Approved: Frank Hadsell
Frank Hadsell,
Thesis Advisor

Golden, Colorado
Date 24 April 84

P R Romig
Phillip R. Romig,
Head of Department
of Geophysics.

ABSTRACT

Recently, an iterative stack algorithm has been introduced by Naess (1979, 1981, 1982). The objective of this paper is to appraise its performance. It was checked in CDP-stack and in velocity analysis; both proved this is a powerful noise suppression tool.

After realizing its effectiveness, investigations were made to see its compatibility to other noise suppression tools. It seems non-compatible with the diversity stack or f-k filtering; which, should be avoided when the iterative stack is used.

Wiener deconvolution could not achieve its goal when executed after the iterative stack. Which infers, when possible, linear processes should precede non-linear processes.

Muting and manual editing will still benefit the iterative stack. In real data, a proper number of iterations should be carefully chosen to get the best enhanced signal.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Figure Page

Table

ACKNOWLEDGMENTS

I would like to express my sincere thanks to my thesis advisor Dr. F. A. Hadsell who has provided helpful suggestions throughout this work, and to Mr. F. H. Ngian and Dr. W. A. Schneider who guided me through graduate studies as my committee members.

For financial support in form of a grant during the year of 1979, I am thankful to the National Science Council, Republic of China. For support in form of a graduate research assistantship from 1983 to present through the Geophysics Department of Colorado School of Mines, I am grateful.

My thanks also goes to my wife Kwaydong, who has given me moral support when I needed it. Last, but not least, I would like to thank my fellow Geophysics graduate students who offered constructive criticism and encouragement.

1. INTRODUCTION

A stack is a composite record made by mixing traces from different records. Examples in seismic exploration are ground mixing, instrument mixing, vertical stacking, uphole stacking, common depth point (CDP) stacking, commom offset stacking, coherency filtering, diversity stacking, migrating by summation method, etc. (Sheriff 1982). The purpose of "stacking" is to enhance the signal-to-noise ratio in the final stacked trace.

The stacking algorithms commonly used in the above processes can be classified into five categories: straight summation, weighted summation, summation after coherency measure, summation after optimum filtering, and non-linear stacking methods. We have a brief review of these algorithms in the next chapter.

Recently, a non-linear iterative stacking algorithm has been introduced and described by Naess (1979). The algorithm has been shown to be adaptable to CDP stack and velocity analysis (Naess and Bruland 1981, and Naess 1982). The primary objective of this paper is to check the performance of the iterative stack in these applications, and to find way to put the iterative stack into routine seismic data processing procedures (All computer

programming work of this paper is assembled in Appendices).

2. ALGORITHMS REVIEW

A quick review of stacking algorithms will be shown below in this order: straight stack, weighted stack, stack after coherency picking, stack after maximum likelihood processing, stack after optimum decon filtering, median selection, and iterative stacking.

2.1 Straight Stack

The same weight is attached to each sample entering the straight stack. The process of straight stack of CDP NMO-corrected data has been well known since Mayne (1962). The enchancement of the S/N ratio by straight summation was an intuitive end result to him at that time. Robinson (1970) gave a quantitative argument that $\sqrt{n}$ is the improvement of the S/N rms amplitude ratio for the straight summation from n traces if all traces have identical S/N ratios and identical signal amplitudes.

Straight stack is still the simplest version accepted as standard routine processing for seismic data. The disadvantage is that small but consistent signals may easily be overruled by a few large noise pulses.

2.2 Weighted Stack

S/N ratio will decrease with increased offset beyond some minimum offset. George et al (1968) suggested that a calculated factor based on offset distance should be multiplied with each trace before stacking.

Statistically optimum stacking was first proposed by Robinson (1970), a theory for weighting seismic records in the stacking process was developed from a statistical model, in accord with that model, a seismic record is described by its signal scale and its S/N energy ratio.

The following statistical properties are imposed on the signal and noise components within a simple seismic window by the model:

1) the signal is identical (except the scale) and correlated on all traces,

2) the noise on any given trace has zero mean and is not correlated to the noise on any other trace; and

3) the S/N ratio is reasonably constant for each individual trace over the time extent of the window.

The mathmatical expression for the jth sample of the ith seismic trace in a gather consisting of I CDP traces is

$$t_{ij} = a_i \ (\ s_j \ + \ n_{ij} \ ), \quad i=1,2,\ldots,I \qquad j=1,2,\ldots,J \qquad (1)$$

Where a_i is the signal amplitude factor or scale (arbitrarily, $a_1 = 1$), s_j is the signal component, n_{ij} is the noise component, and J is the number of discrete samples per trace. The signal energy and the noise and total energies respectively for the ith trace are defined as follows:

$$S = \sum_{j=1}^{J} (s_j)^2, \quad N_i = \sum_{j=1}^{J} (n_{ij})^2, \quad T_i = \sum_{j=1}^{J} (t_{ij})^2 \qquad (2)$$

The unique S/N energy ratio r for the ith trace is defined as

$$r_i = \frac{S}{N_i} \qquad (3)$$

The objective of the optimum stack is to determine a weighting factor w_i to be applied to the corresponding ith trace such that the S/N ratio of the trace resulting from the weighted stack will be maximized.

The optimum stack for I traces is achieved by a logical extention of the two-fold weight stack: (see Robinson 1970, equations 4 to 17)

$$(t_j^{(I)})_{\text{optimum}} = \frac{1}{c} \sum_{i=1}^{I} \frac{r_i}{a_i} t_{ij} \qquad (4)$$

where c is a normalization factor. Stated in words, the

optimum stack is effected by first equalizing the signal scale on each trace, then weighting the resulting traces with their respective signal-to-noise energy ratios, and finally, stacking and normalizing these traces.

To implement the optimum stack, we should know the scale and S/N ratio for each trace in the stack. We cannot determine these values in practice. The statistical estimations are based on the simple trace-cross-product sums defined as

$$x_{mn} = \sum_{j=1}^{J} t_{mj}\, t_{nj} \quad , \qquad (m,n=1,2,\ldots,I) \tag{5}$$

Robinson (1970, equations 21 to 24) demonstrates that the best approximations to the trace scale and the S/N ratio for trace n in I traces are:

$$a_n = \frac{1}{(I-2)} \sum \frac{x_{nk}}{x_{1k}} \quad , \text{ where } k \neq n$$

$$r_n = \frac{1}{(I-1)} \sum \frac{1}{\frac{x_{mm}x_{nn}}{x_{mn}} - 1} \tag{6}$$

The computation or these is very time-consuming. If we assume all S/N ratios are approximately equal (Robinson 1970, scheme IV), then approximately,

$$w_n \propto \left(\frac{1}{x_{nn}}\right)^{0.5} \quad , \quad n=1,2,\ldots,I \; . \tag{7}$$

This is a mucn simpler weighting function which merely effects a normalization of each seismic trace on its total energy. These weights are used in what is called the diversity stack.

In routine processing, bad traces are first deleted by manual editing . This gives bad traces a zero weighted value and assures that the S/N ratio of each trace falls in an acceptable range. The traces are then stacked by diversity stack. This simple combined process has been phenomenally consistent at achieving a notable improvement in the stack (Ngian, 1983).

2.3 Stack after Coherence Picking

Coherency stack combines data which satisfy certain trace-to-trace coherence criteria. Many coherence measures were reviewed by Neidell and Taner(1971). The commom objectives of coherency measure include signal prediction, detection of shifts, and extraction of common signal. Signal in the channels can be measured, enhanced, shaped, or even ignored after the use of some coherence measure. It is too broad a subject for discussion here.

The coherency measure used in stack is largely dependent on the coherency criteria used. Coherence picking may be better than the manual editing in some sense, and the result of the coherent stack might be slightly better than that of a routine weighted process.

2.4 Stack after Maximum Likelihood Filtering or Optimum Deconvolution Filtering

Optimum stacking filters and their performance in suppressing uncorrelated noise has been investigated by White(1977). He defined the "optimum stack filtering" in terms of two processes; the "maximum likelihood process" which minimizes the output noise power under the restriction of no signal distortion, the "Wiener deconvolution process" which minimizes the mean square deviation or the output from some arbitrarily selected "desired" output.

The basic principles or maximum likelihood processing are still contained in the theory for the weighted stack, and the maximum likelihood weights w_i for the trace i or I CDP traces are obtained by:

$$w_i = \frac{1}{a_i} \cdot \frac{r_i}{\sum_{j=1} r_j} \qquad (8)$$

where a is the signal scale and is the S/N power ratio (see White 1977, equations 1-7).

Allowing for this leads to the case of optimum stacking filters, the above equation must be valid at every frequency in the signal

$$w_i(f) = \frac{r_i(f)}{a_i \cdot \sum_{j=1} r_j(f)} \qquad (9)$$

Thus if, say, 17 frequencies between zero and the Nyquist frequency are allowed for the definition of the stack filters, the S/N ratios at these 17 frequencies must be evaluated to obtain the weights $w_i(f)$ from the above equation. The frequency response $w_i(f)$ can then be Fourier transformed to produce a 33 point stacking filter for trace i.

White (1977) concluded that optimumly filtered stack is only slightly better than the straight stack. The improvement or S/N ratio of the straight stack will usually be at least eighty per cent of that of an optimumly filtered stack. Moreover, the straight stack suppression of uncorrelated noise is best at low S/N ratios where it is most needed.

The multichannel Wiener filtering, on the other hand, actually requires only single channel operations. According to Green, Kelly and Levin(1966), Wiener multichannel filtering is essentially equivalent to the application or a single channel Wiener filter to the output of a maximum likelihood process. The later is more computationally effective than the former.

In fact, no matter whether one applies the maximum

likelihood filtering or the multichannel Wiener filtering, the improvement of the output will be restricted by the design parameters which must be obtained from the seismic data itself, and the best result is at most the desired trace (Ngian, 1983). This might be the reason the optimum stacking filters have not been widely used in the routine seismic data processing.

2.5 Median Stack

The median stack algorithm is to select only one "median-value" instead of combining all data. In the selection process, all data contributing to the stack are line-up in an order from high to low, and the one located in the middle will be chosen as the representative value (Ngian 1983).

This method is not linear and tends to change the frequency content in an unwanted manner (Naess 1979). It might be able to suppress the noise when a few large noise bursts overrule the small consistent signals. However, the iterative algorithm introduced in the next section will do a better job for the same price.

2.6 Iterative Stack

This algorithm has been first introduced and described by Naess(1979). The algorithm is expressed mathematically below by an example of CDP stack. The input to the algorithm consists of all amplitudes at one reflection time-level of the NMO-corrected CDP-gather. Parameters are defined as follows:

a_i^k =positive amplitude number i after k-th iteration,

b_j^k =negative amplitude number j after k-th iteration,

m=number of positive amplitudes,

n=number of negative amplitudes,

M=m+n (note that Naess(1979) used M as a parameter),

S+=sum of positive group divided by M,

S-=sum of negative group divided by M.

The 1st-iteration:

$$S_+^1 = \frac{1}{M} \cdot \sum^m a_i^1$$

$$S_-^1 = \frac{1}{M} \cdot \sum^n b_j^1$$

$$S^1 = S_+^1 + S_-^1 = \frac{1}{M}\left(\sum_{i=1}^m a_i^1 + \sum_{j=1}^n b_j^1 \right)$$ --1-iteration output,

the 2nd-iteration:

if $a_i^1 > S_+^1$ set $a_i^{(2)} = S_+^1$

$a_i^1 \leq S_+^1$ $a_i^{(2)} = a_i^1$

$b_j^1 < S_-^1$ $b_j^{(2)} = S_-^1$

$b_j^1 \geq S_-^1$ $b_j^{(2)} = b_j^1$

then $S_+^{(2)} = \frac{1}{M} \cdot \sum^m a_i^{(2)}$

$S_-^{(2)} = \frac{1}{M} \cdot \sum^n b_j^{(2)}$

$S^{(2)} = S_+^{(2)} + S_-^{(2)}$ --2-iteration output,

The higher-iterations just repeat the computation above, the q-th iterative stack is

$$S_+^q = \frac{1}{M} \cdot \sum^m a_i^q$$

$$S_-^q = \frac{1}{M} \cdot \sum^n b_j^q$$

$S^q = S_+^q + S_-^q$ --q-iteration output.

It is noted that the output of the 1st-iteration is equivalent to the result of straight stack.

This is a procedure where the difference between each input trace and the mean value is subtracted from the input trace; the separation of the positive and negative group is to make the subtracting procedure convergent. To put it another way, if the samples are not separated the process that causes the positive values to converge will cause the negative values to diverge.

The iterative algorithm could be considered as a combination of two algorithms:

1) an algorithm to find the sum of the most consistent positive amplitudes and the most consistent negative values,

2) an algorithm to enhance the most consistent polarity; for example, if all amplitudes are equivalent in absolute value, the computation formula could be represented by

$$S^q = (m/M)^q A + (n/M)^q B$$

where $a_1 = a_2 = \ldots = a_m = A$

$b_1 = b_2 = \ldots = b_n = B$

which could be named as "higher degree stacking".

Generally the primary amplitudes should be equal in strength and polarity, and all noise amplitudes shall be random or less consistent than primary ones. Thus, after several iterations the primary amplitudes will be emphasized

since the noise is suppressed more quickly.

In order to illustrate the iterative process, we made a test computation on a set of 10 random numbers as shown in figure 2-1. The initial 10 random numbers range between -0.5 and +0.5, six of them are positive. Since neither the positive group nor the negative group are dominant, both groups were suppressed quickly by iterative process. Negative group dropped to the zero level first (after 5 iterations) because they are relatively inconsistent with respect to the positive group. After 10 iterations the positive group was also exterminated.

Another test was made on a set of 10 positive random numbers which range between 0.00 and 1.00 (figure 2-2). The process was made up to 20 iterations. It clearly demostrates the rate that data values were changing; when more data were falling near the same value (more consistency appeared) the slower their values would be decreased. This explains how the iterative stack sustains the small but consistent signals.

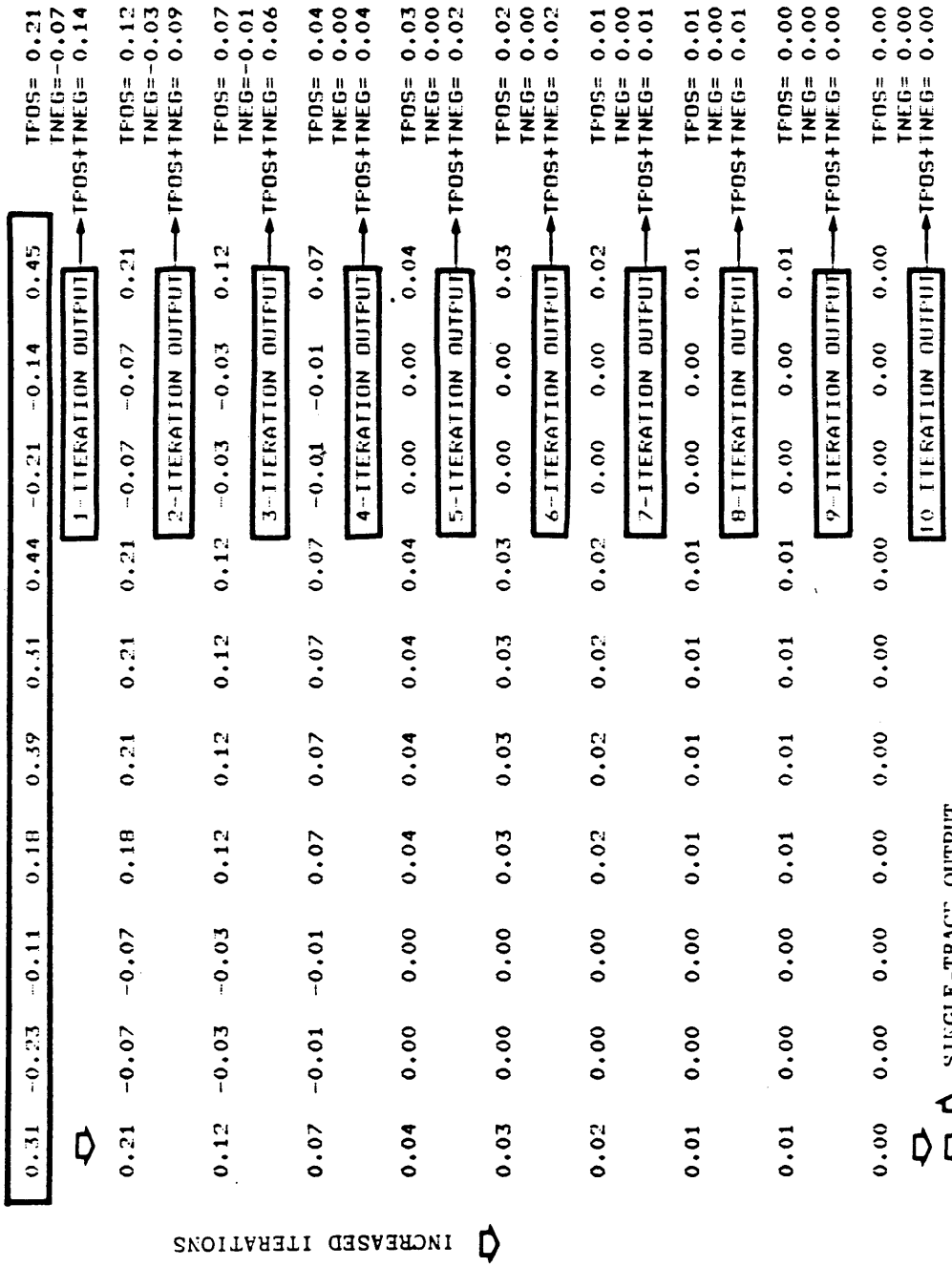


Figure 2-1. Iterative algorithm tested by 10 random numbers.

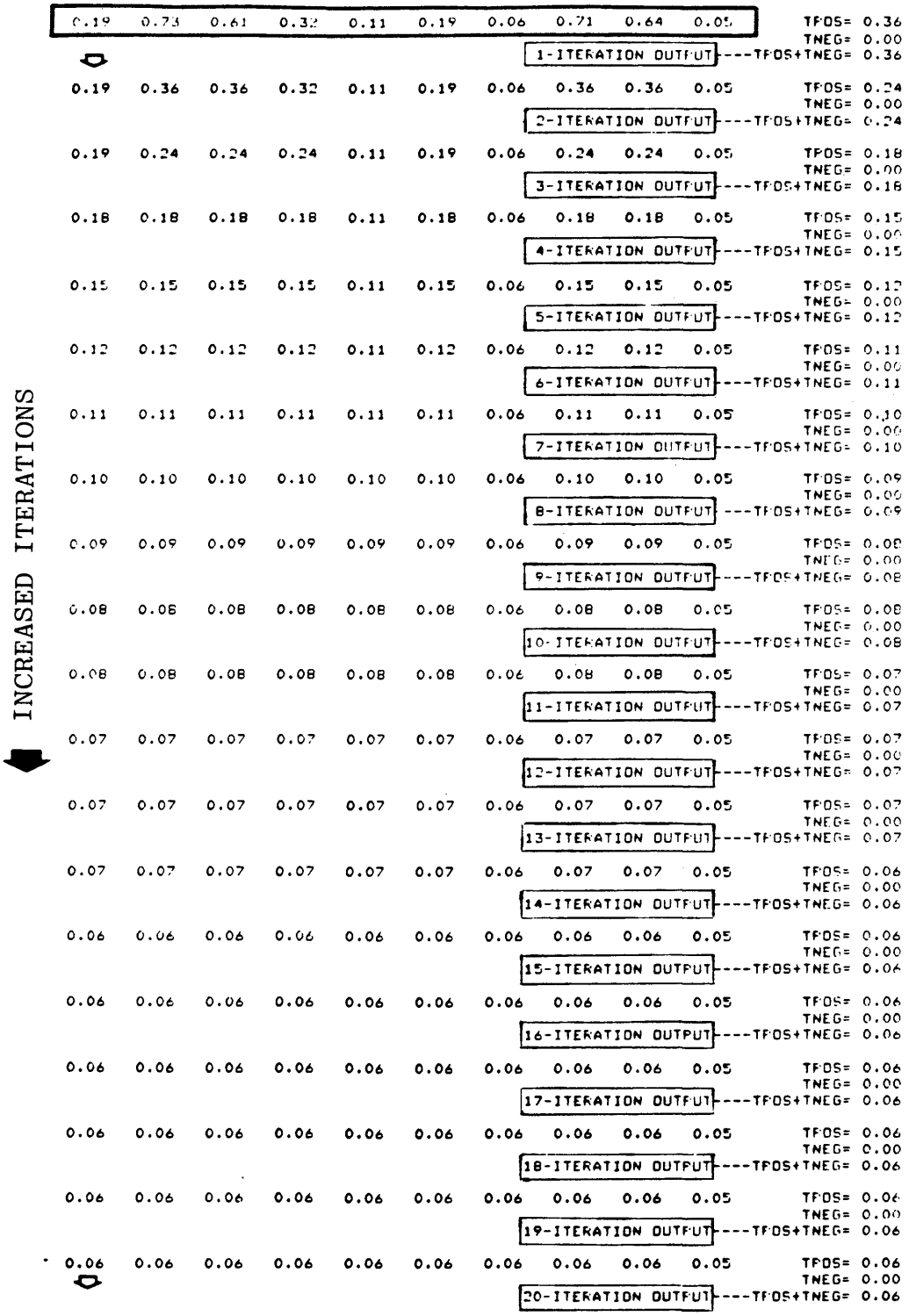


Figure 2-2. Iterative algorithm tested by 10 positive random numbers.

Summary of the algorithm review:

(1) Straight summation is still the conventional routine stack algorithm, but actually it is an "optimum" weighted stack because manual editing and the diversity stack are usually added.

(2) Coherent picking might be better than the manual editing, but it is largely dependent on the coherency criteria used.

(3) Multichannel optimum filtering is unlikely to be as popular as straight stack, it is not adapted to routine multichannel processing.

(4) Iterative and median stack are non-linear processing, and hence change the frequency content.

3. SYNTHETIC EXAMPLES OF THE USE OF THE ITERATIVE ALGORITHM

We are going to use a simple model to check the performance of the iterative algorithm in the stacking of multichannel data, the algorithm will also be examined with background noise. The same model will be used to illustrate the performance of the iterative algorithm in single trace processing. A second model will be used to show the filtering effect of the iterative algorithm in velocity analysis. Finally we will introduce more realistic modelling for comparison and discussion.

3.1 Naess' Model- Example of Iterative Stack

A 3-layer earth model and its parameters is shown on figure 3-1. For this model I generated the 12-fold CDP-gather shown in figure 3-2. Each trace consists of a waterbottom reflection, two primaries, and five direct waterbottom multiples. The waterbottom reflection and all multiples were given the same strength. The primary-to-multiple amplitude ratio is assigned 1:10. The basic wavelet used in the model is shown in Appendix A: The program 'TRPLOT'. These model parameters are similar to

MODEL PARAMETERS

sample interval: 2 ms
primary-multiple amplitude ratio: 1:10
recording in 12-fold
minimum offset 100 m, maximum offset 650 m.
the model consists of waterbottom reflection,2 primarys,
and 5 direct waterbottom multiples.

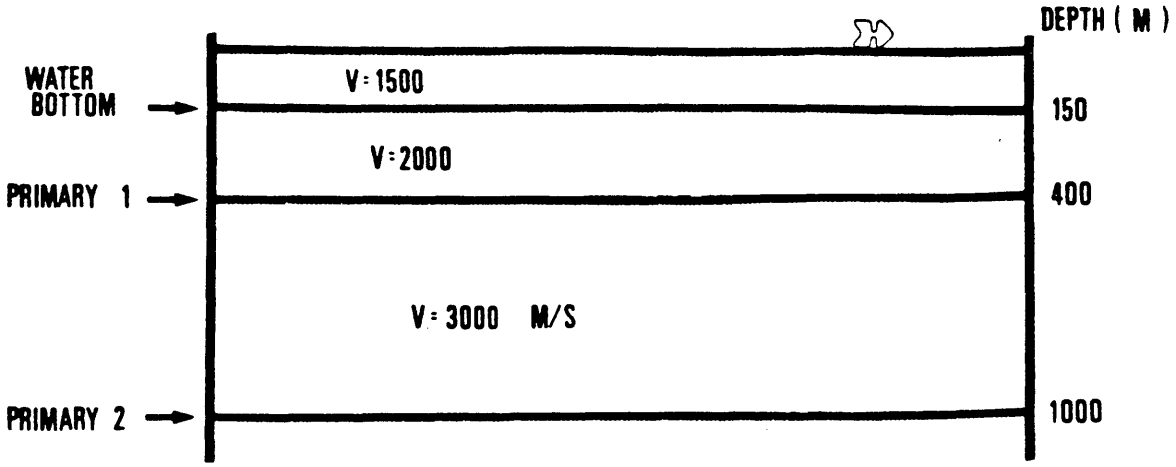


Figure 3-1, A 3-layer model and its parameters.

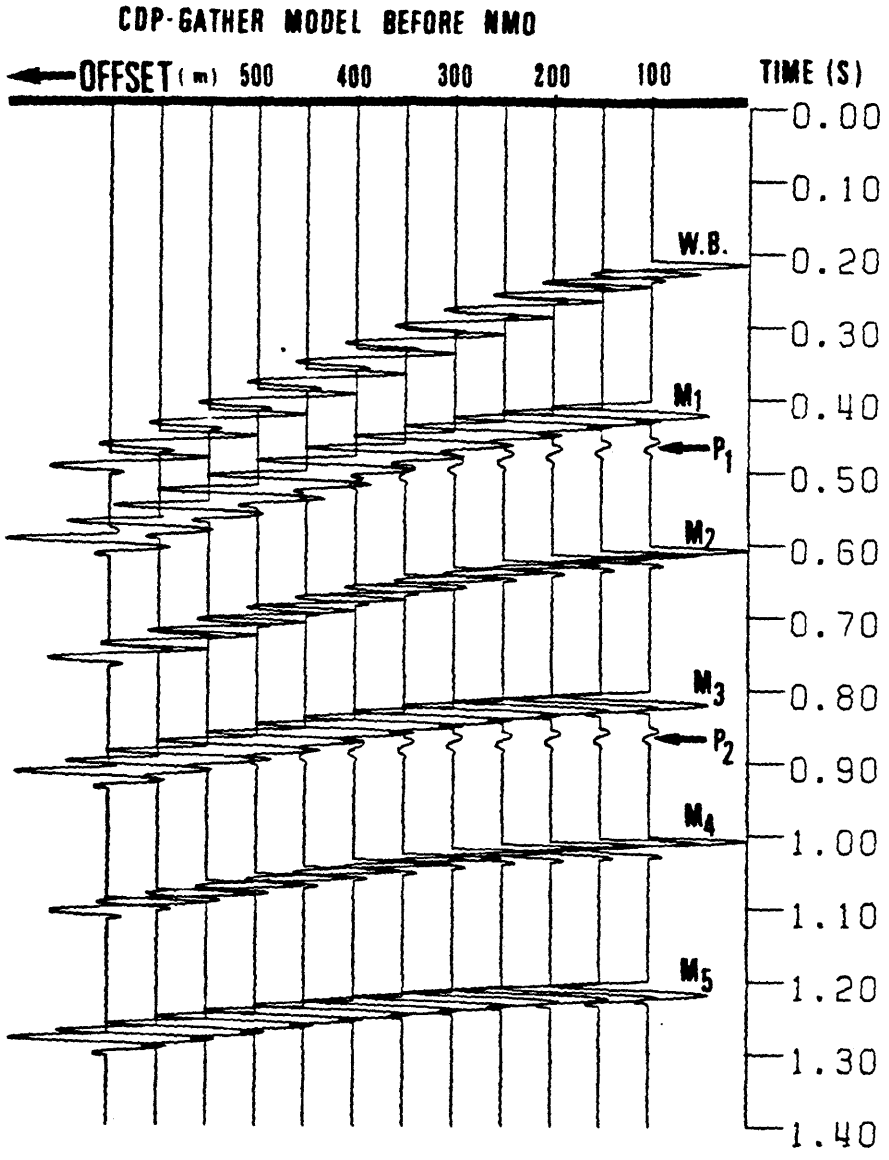


Figure 3-2, CDP-gather generated from the 3-layer model.

Naess' model (Naess 1979) and provide simplified reflections for easy observation. The realism of such data, as a model of a preprocessed gather, is discussed later in this chapter.

Figure 3-3 shows the CDP-gather traces after NMO-correction, the streching effect is significant in far offset traces. Figure 3-4 displays the resulting final stacked traces after applying the iterative algorithm from 1-iteration to 12-iterations. The 1-iteration version is the same as straight stacking.

All the multiples are more and more reduced as the number of iterations increases. After five iterations the primary reflectors already dominate the trace. This test proves that the iterative stacking is a powerful tool and the 5-iteration or 6-iteration version is probably adequate for the display of the final seismic section.

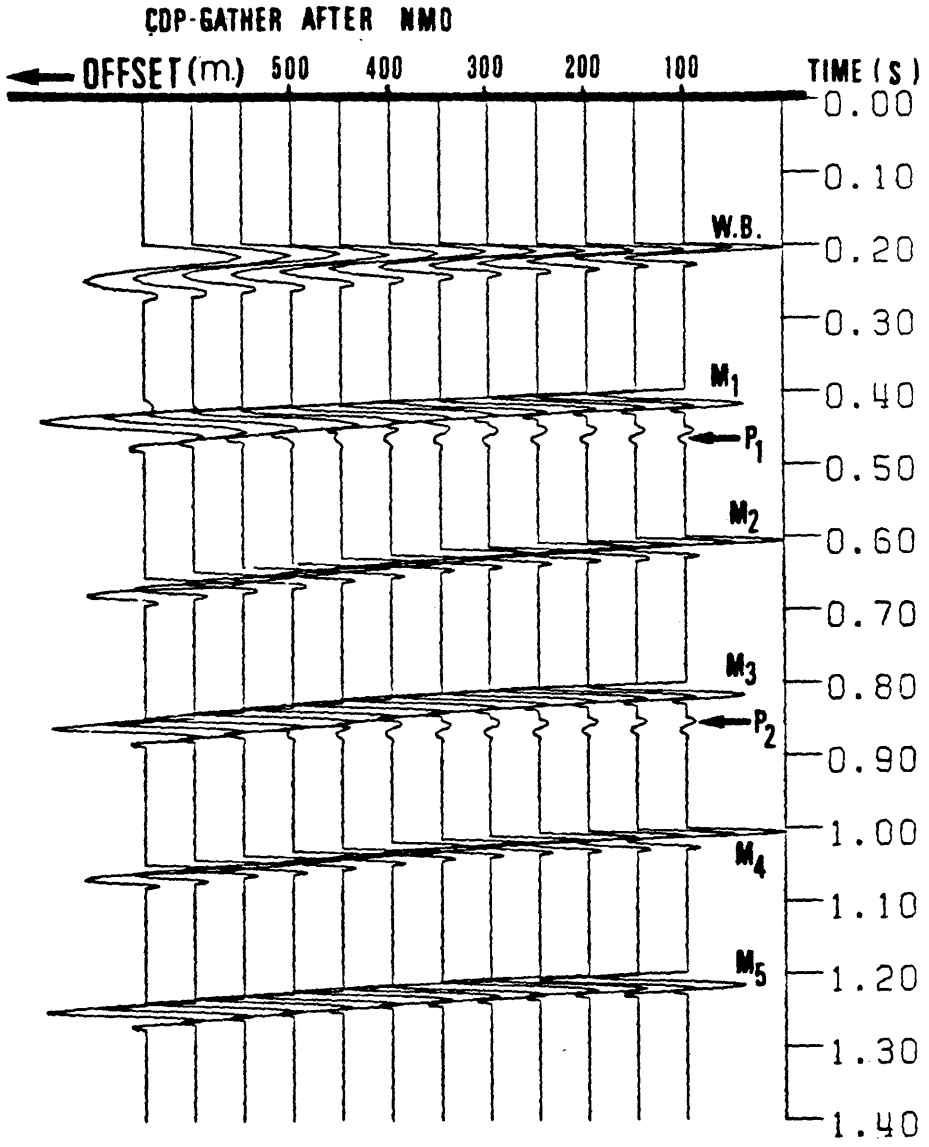


Figure 3-3, CDP-gather of figure 3-2 after NMO-corrected.

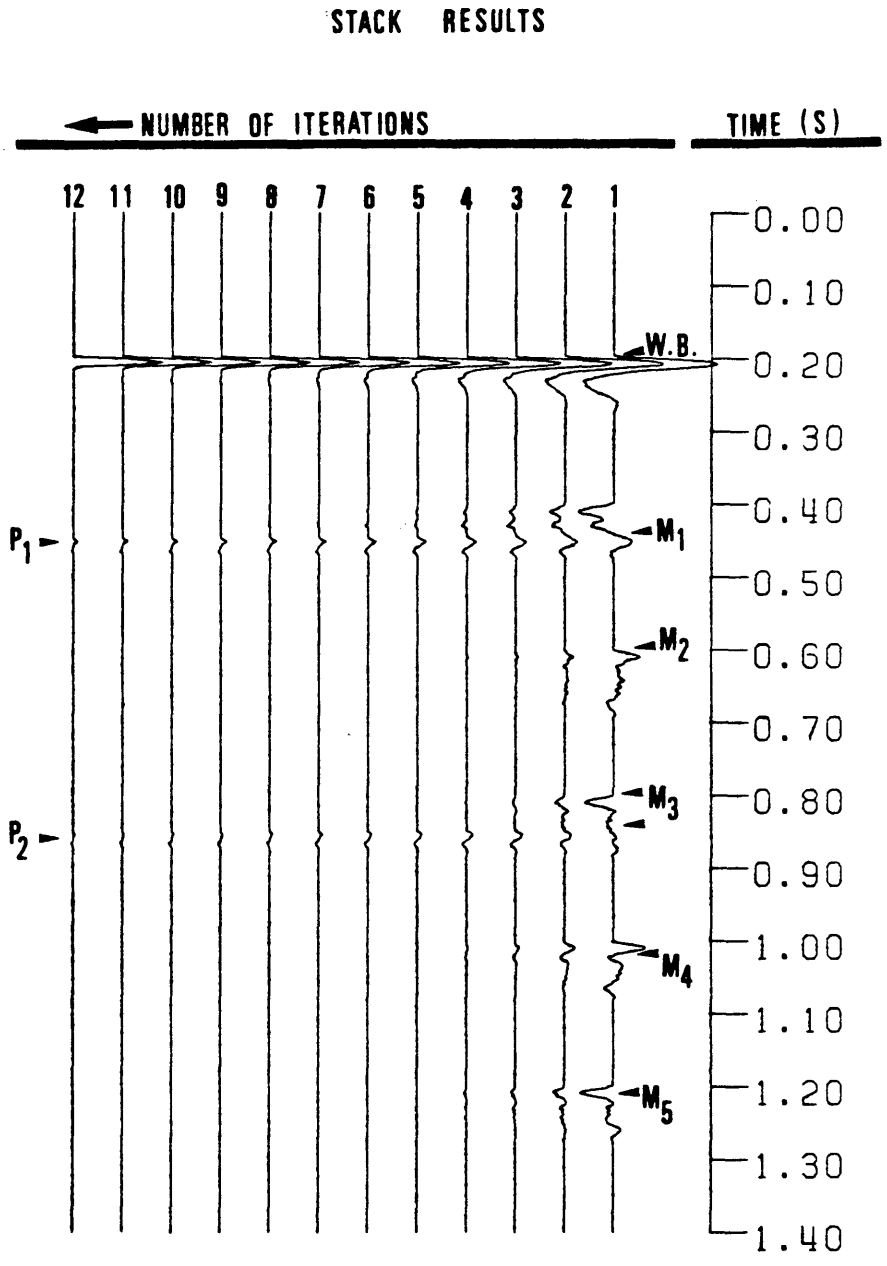


Figure 3-4, Result of iterative stack from figure 3-3.

3.2 Naess' Model with Background Noise Involved

It is interesting to know the performance of iterative stacking when background noise is involved in the model. The background noise is generated as filtered random numbers and added to each trace, the signal-to-noise amplitude ratio is about 5:1, where the signal value is calculated from the average of the three primaries, and the noise value is calculated from the average absolute value over the whole trace.

Figure 3-5 shows a 12-fold CDP-gather from the previous model with random noise added. Figure 3-6 shows NMO-corrected traces, and figure 3-7 displays the resulting final stack trace after 1-iteration to 12-iterations. Most of the background noise is removed from the stacked traces after 6 iterations. It could be stated that as long as the noise is not changing the consistency of the signal, as would be the case given a statics problem, the noise will be cleaned out of the section while the signal stays.

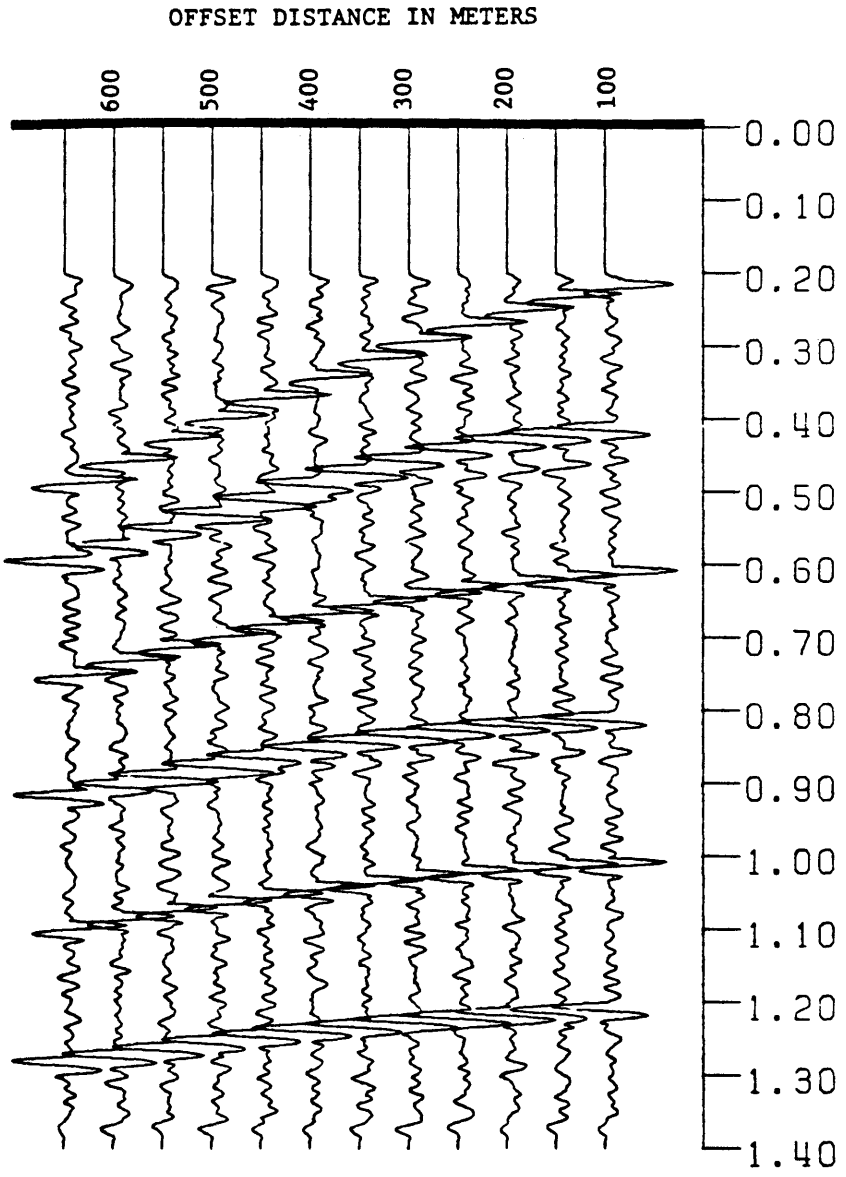


Figure 3-5, CDP-gather generated from 3-layer model with background noise involved.

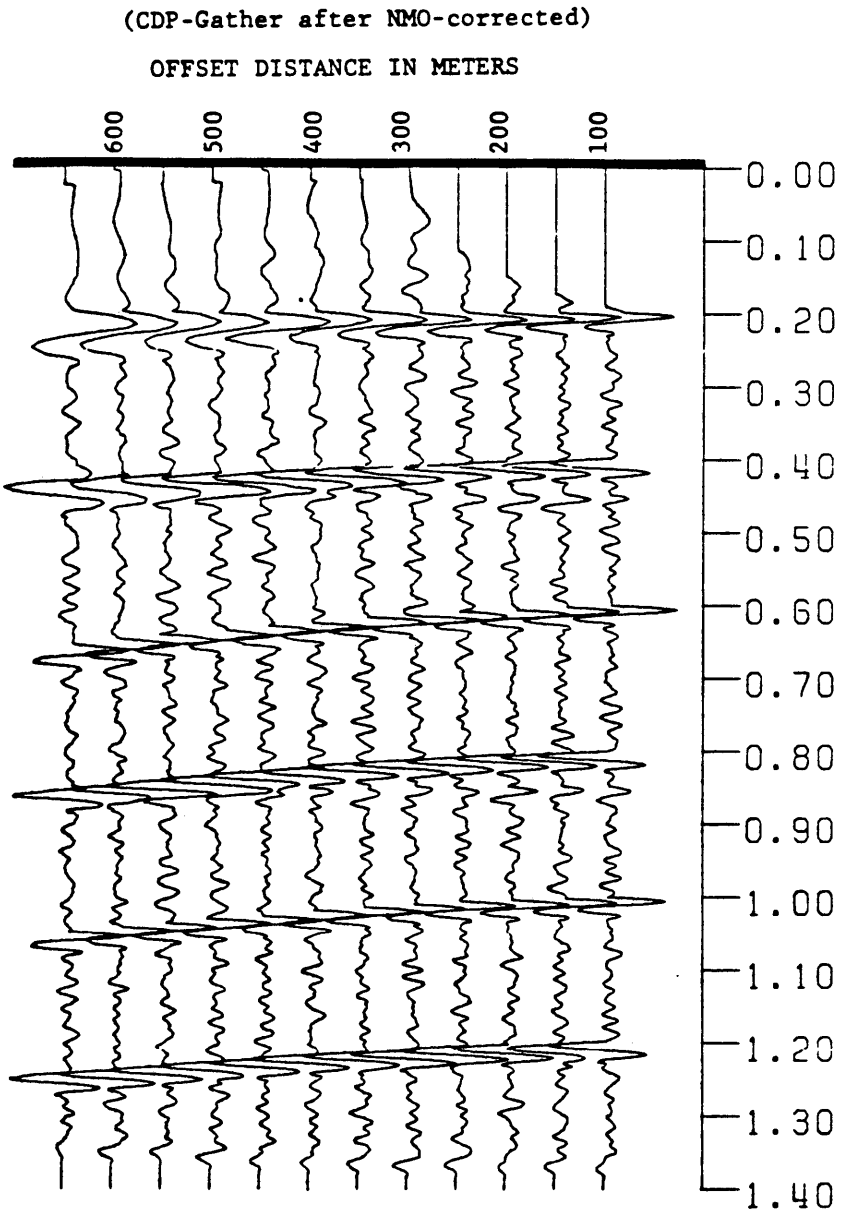


Figure 3-6, CDP-gather of figure 3-5 after NMO correction.

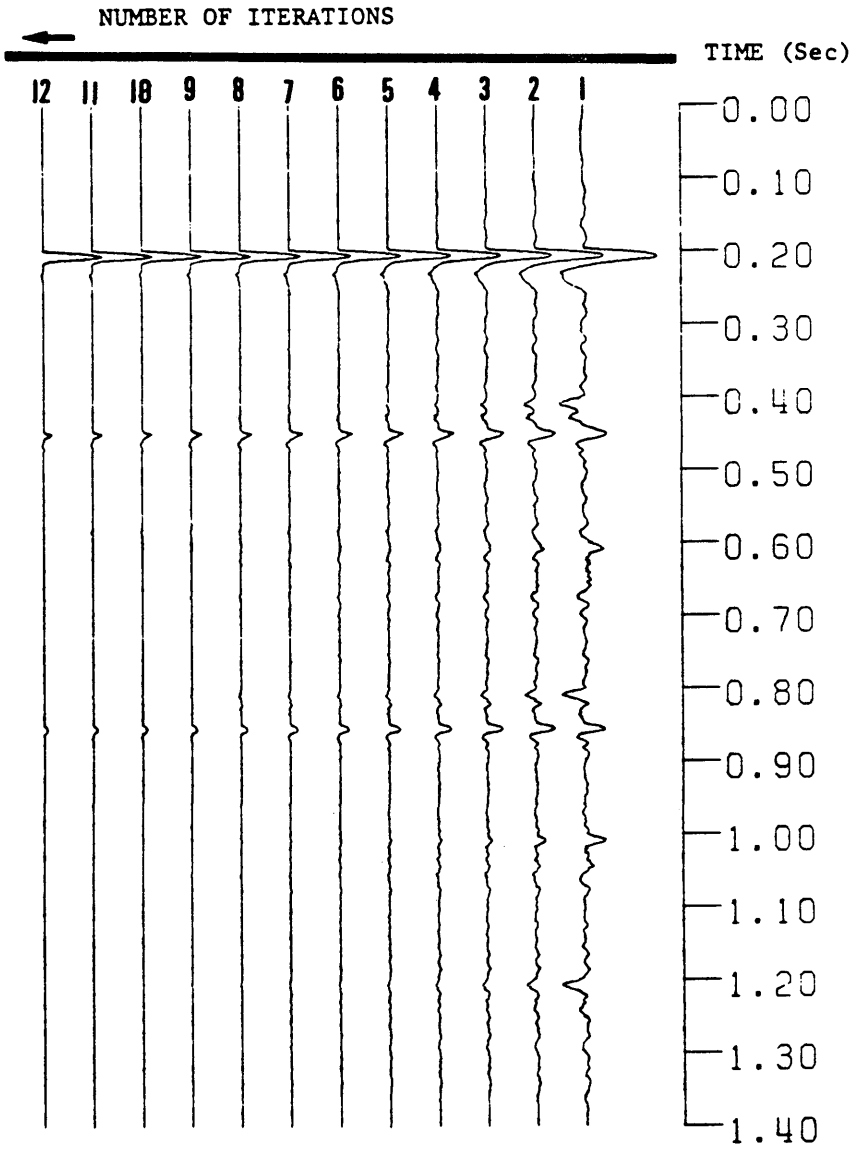


Figure 3-7, Result of iterative stack from figure 3-6.

3.3 Example of Single Trace Process

High resolution has always been a goal of the exploration industry. The maximum resolution in seismic reflection work would be obtained by using a point-source and recording the reflected signals with one single detector located as near to the source as possible. But the conventional CDP-stacking, whose sole object is to improve the signal-to-noise ratio and attenuate multiples, implies straight summation of signals recorded at distances up to 2.5 km from the source, there is a contradiction in using the CDP-method when high resolution is required.

The resolution in seismic reflection data may be subdivided into lateral and vertical resolution: the lateral resolution is concerned with the ablity to detect small changes horizontally from trace to trace. In CDP-stacking, if the lateral extension of the reflecting area is larger than the extension of the change we want to detect, then the stacked data have a tendency to hide such a change. For example, a small fault may appear as a slight weakening in a reflector instead of a clear break.

The vertical resolution is concerned with minimum distance between two reflectors which is necessary in order to detect them as two separate events. The CDP-stacking

procedure affects the vertical resolution for two reasons: first, the stacking implies summation of traces with large offsets and hence more attenuation. Thus vertical resolution of a stacked trace is less than that of a single trace with short offset. Second, the stretching effect in normal moveout corrections may introduce further differences.

To obtain high resolution it would therefore be best to use only one trace recorded with the least offset for construction of the final seismic section. However, it is obviously also necessary to improve the S/N ratio on the near trace to a degree comparable to a conventionally stacked trace. A way towards achieving both of these goals is the use of iterative stacking with respect to the near trace only, that is called the single trace process (Naess, 1982). In practice, the iterative stacking is done equally with respect to all traces in the CDP-gather, and only the near trace is chosen as the output rather than the sum of the positive and negative groups.

Figure 3-8 displays the resulting final iterative stack with respect to the near trace from 1-iteration to 12-iterations, the 1-iteration shows more high frequency components than the conventional straight stack of figure 3-7, the difference of frequency content is also clearly evident when one compares the waterbottom reflections in the

traces on both figures. Furthermore, single trace processing shows a better ability to keep its original form than the ordinary iterative stack especially in the first few iterations.

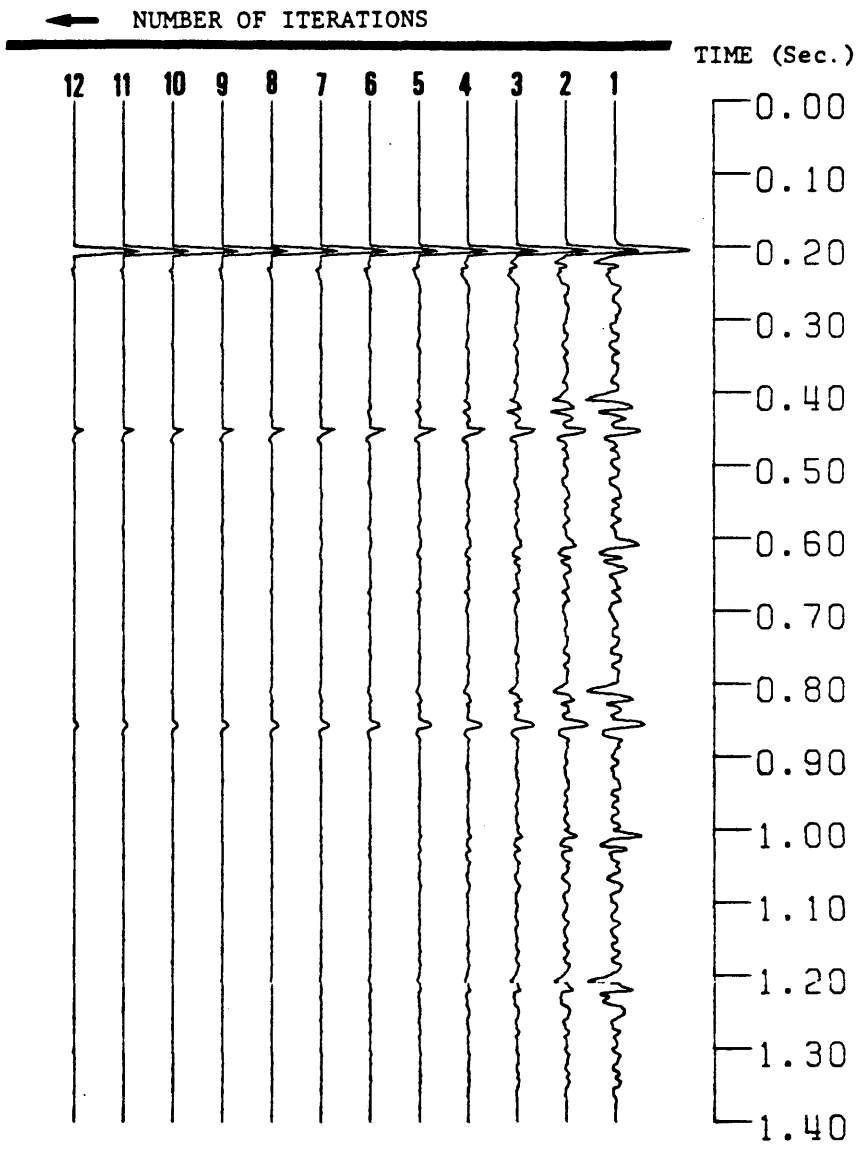


Figure 3-8, Result of single trace process.

3.4 Example of Velocity Analysis

Velocity analysis is one of the important elements in seismic data processing since stacking velocities determine the quality of the final seismic sections. The velocities are also important independently of stacking, since they are related to physical parameters of geological significance.

Methods of velocity analysis are based on a visual or computer coherency measure. Prior to the coherency measurements there is normally no filtering except possibly a deconvolution, and the result of the velocity analysis is therefore highly dependent on the capability of the coherency measure to disregard the disturbing influence of noise events. Coherency measures are often based on different design philosophies (Neidell and Taner 1971), but the assumptions about noise and signal structures are seldom achieved in practice.

If disturbing noise can be removed from the data before the coherency measurements, the quality of velocity analysis will be improved. Naess (1981) has proposed that the iterative algorithm can achieve this filtering job, a check of its performance is illustrated below by a second model.

Figure 3-9 shows a 4-layer model and its parameters,

Synthetic CDP-gather Model
This is based on horizontally layered model with constant parameters within each layer and has 3 primary reflections P1,P2,P3 in addition to the waterbottom reflection, also water confined multiples are included.
recording in 12-fold, minimum offset 200 meters, maximum offset 1300 meters.

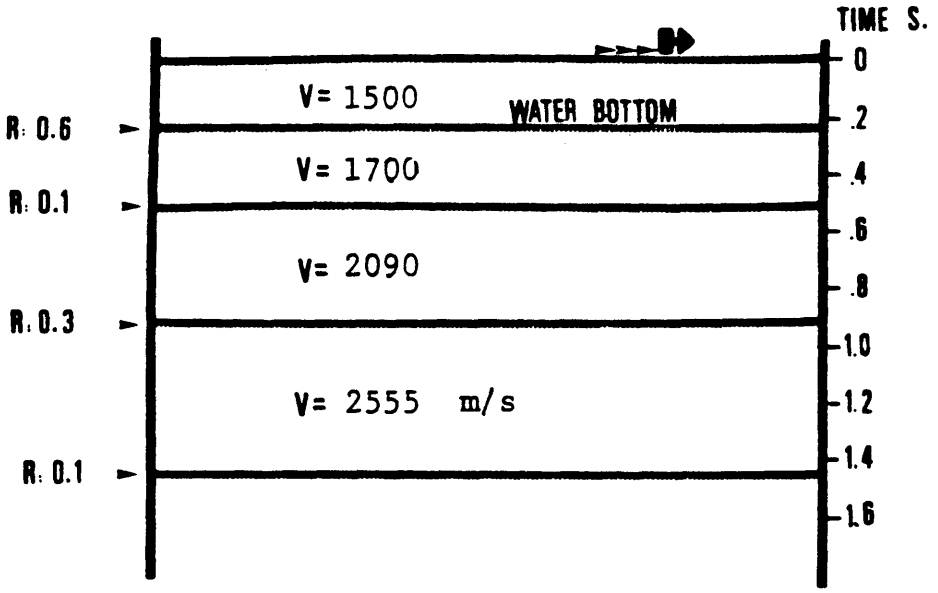


Figure 3-9, A 4-layer model and its parameters.

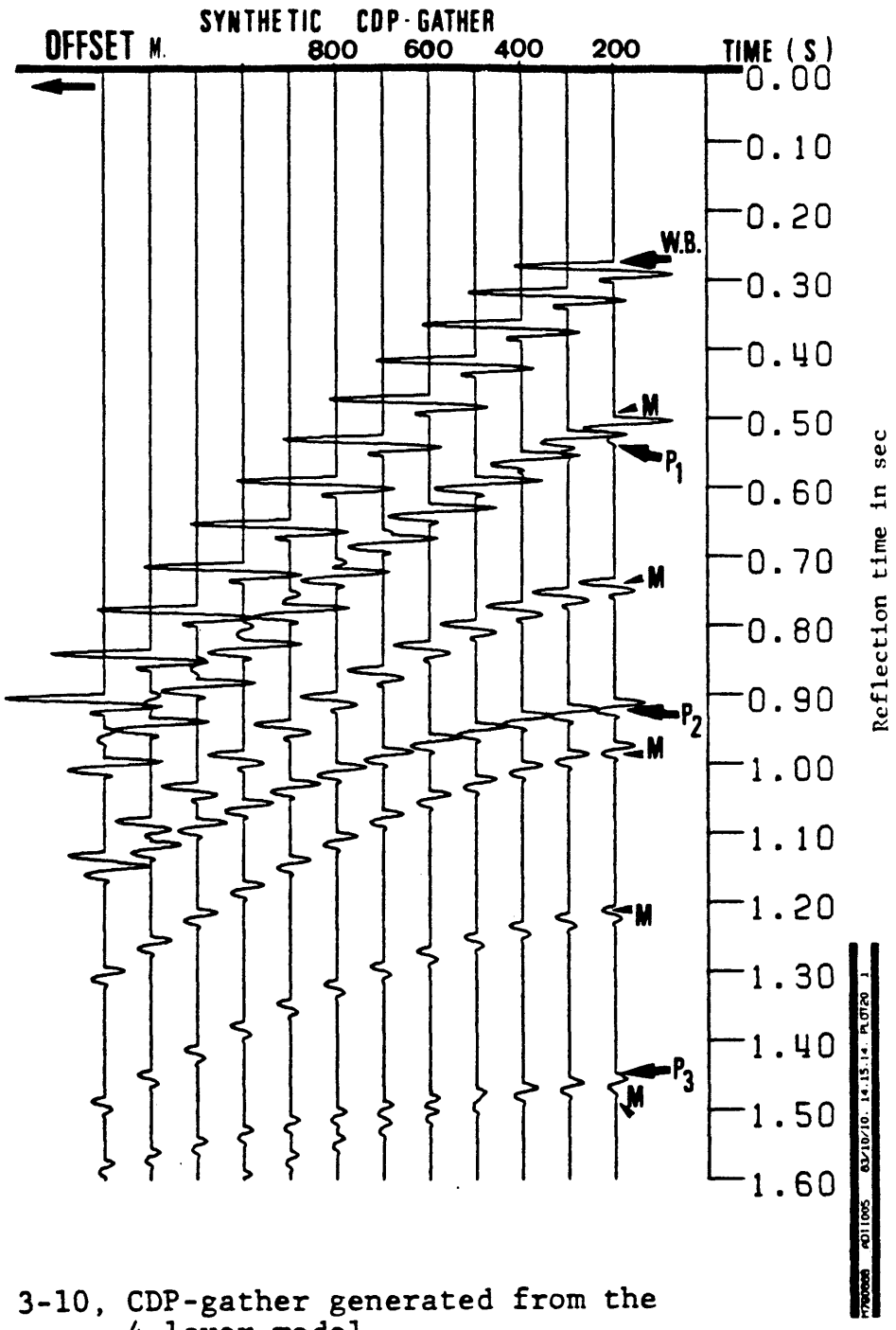


Figure 3-10, CDP-gather generated from the 4-layer model.

the CDP-gather shown on figure 3-10 is generated from this model under the assumption that only water confined multiples are important, these can be easily seen as a sequence of events following the primaries. The coverage is 12-fold, minimum offset is 200 m and maximum offset is 1300 m. The interval velocities used in the model are 1500, 1700, 2090, 2555 m/s respectively from top to bottom. The reflection coefficients used are 0.6, 0.1, 0.3, and 0.1 from water bottom and downwards. (See the next section for a discussion of the realism of the model.)

The method of velocity analysis we chose is the constant velocity stack (CVS) method. A conventional CVS-analysis is shown in figure 3-11, a display of iterative CVS-analysis after 2 iterations is shown on figure 3-12, and figure 3-13 shows an iterative CVS-analysis after 4 iterations. The velocities used range from 1460 m/s to 2240 m/s with an increment of 20 m/s. There is an impressive improvement in the definition of primary events on the iterative CVS versions. The background noise, caused by random line-ups which are present on the conventional CVS, have been effectively suppressed on the 4-iteration version.

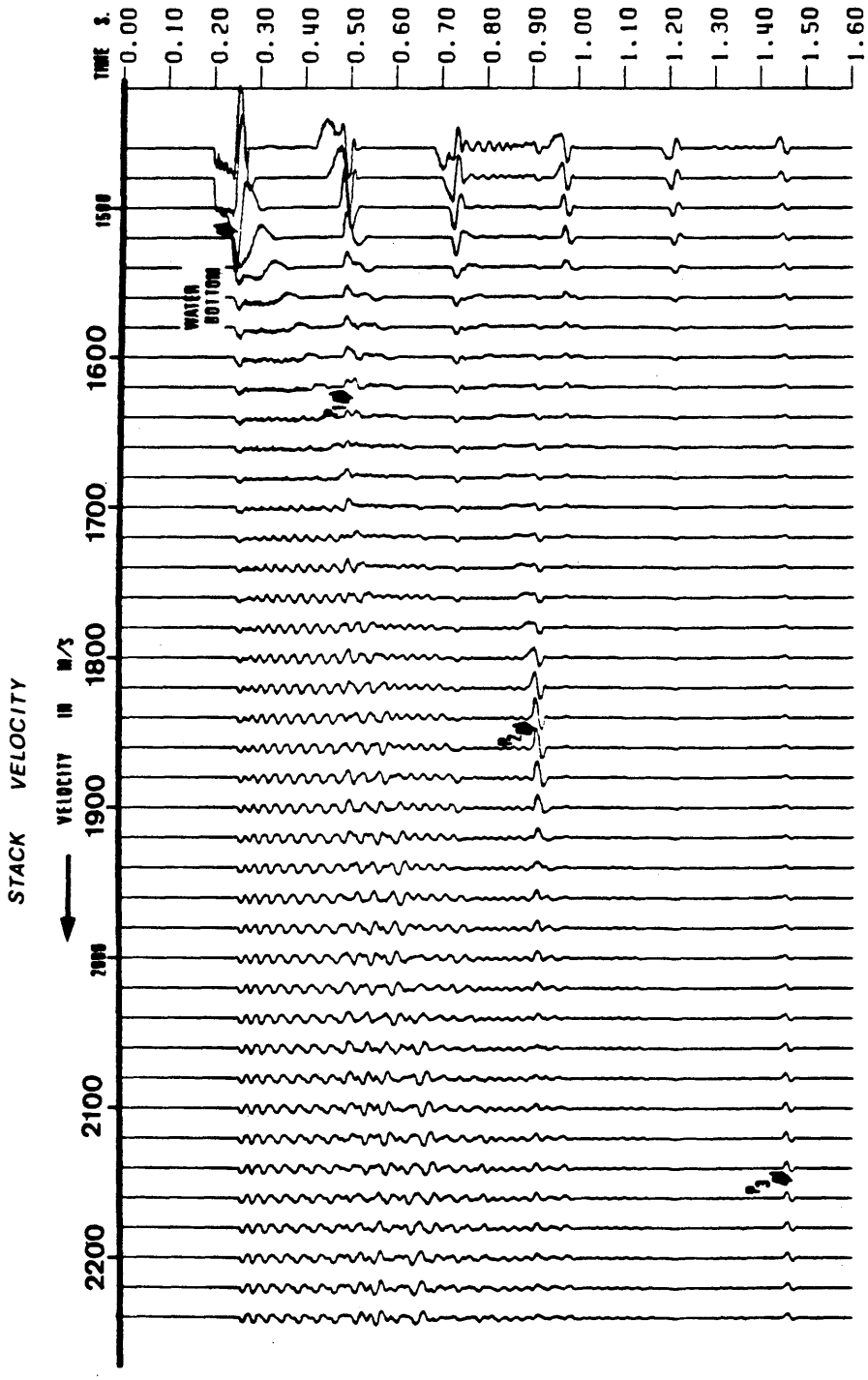


Figure 3-11, Conventional constant velocity stack(CVS).

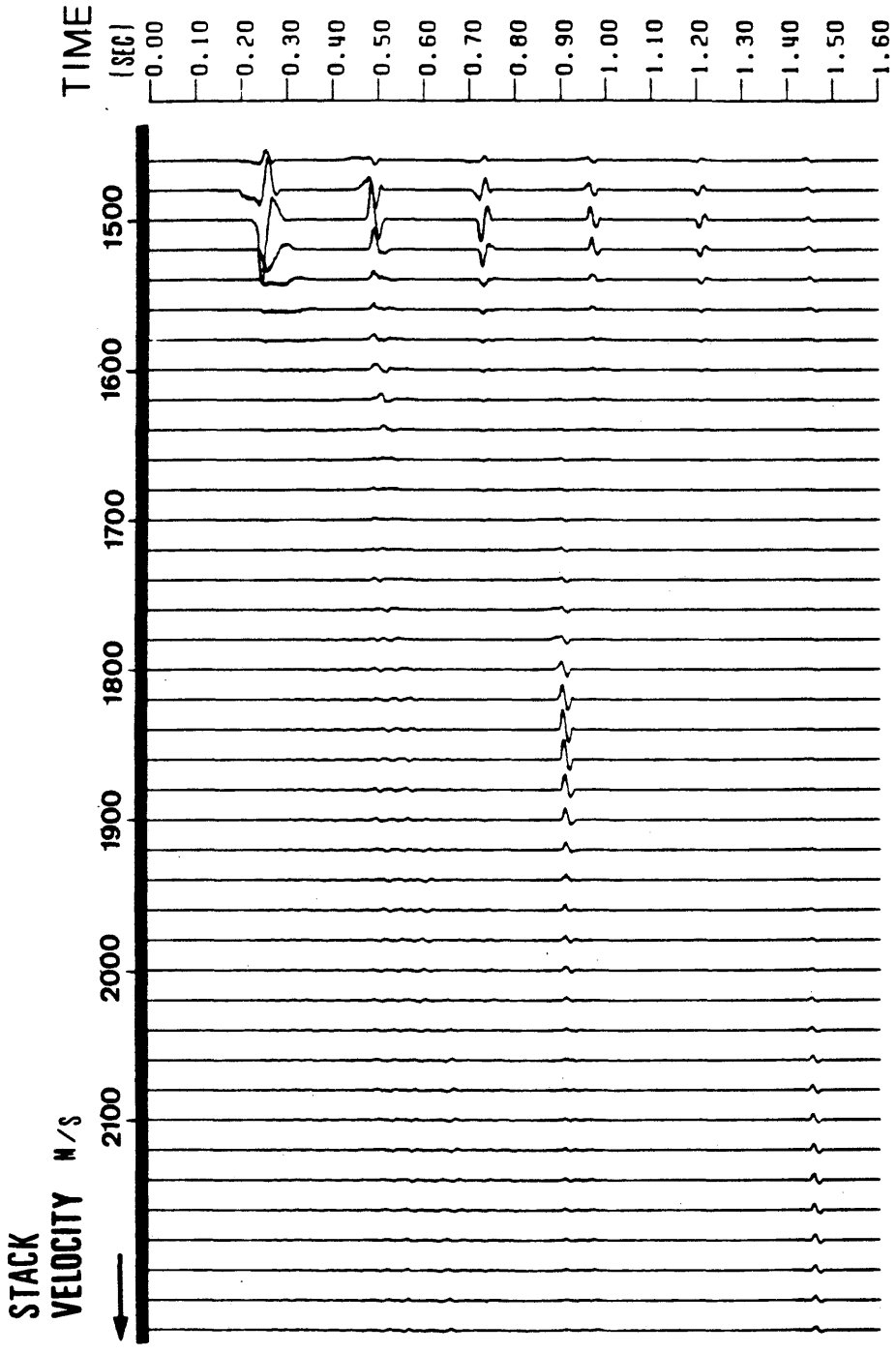


Figure 3-12, Iterative CVS-analysis after 2 iterations.

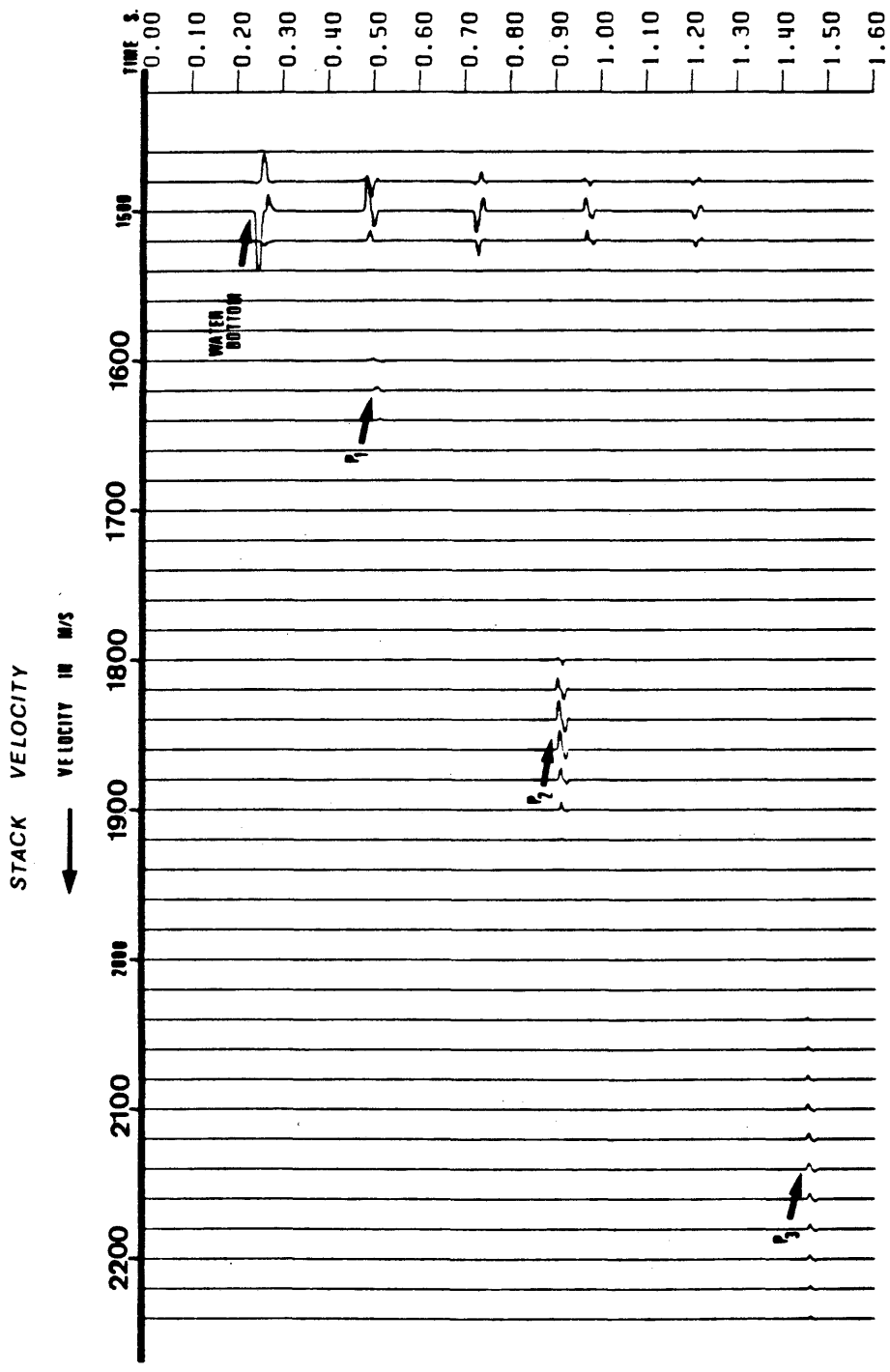


Figure 3-13, Iterative CVS-analysis after 4 iterations.

3.5 A Discussion with More Realistic Modelling Work

In the previous two modelling cases, the CDP-gather generated from the models were simplified in that only several easily observed raypaths were counted and reflection coefficients were not computed from the velocities. In order to justify these assumptions, a more realistic modelling suite of programs was applied. These are called the GRAY suite (Generalized RAY) of the Colorado School ot Mines. The output of GRAY for the first model consists or 38 rays for each receiving station as shown in table 3-1, whicn is generated under the consideration that reflections come from both p-wave and s-wave ray-tracing with eight or less legs and 1.5 seconds recording time. The previous modelling work shown in figure 3-2 included only seven or these rays.

More rays means more disturbing events that hide the primaries. Figure 3-14 shows the more realistic CDP-gather from the GRAY suite, in which primaries are disturbed and are not so consistent as those in figure 3-2. The most interfering events are those strong responses from head waves and shear wave conversions in shallow depth as indicated in figure 3-14 and table 3-1.

The other important difference between our model and

the GRAY result is that the output of GRAY has taken into account the fact that the reflection amplitudes are changing with respect to the incidence angles. In the real earth, the changing does happen and the signal responses gathered from different offset angle could not be equal in strength or polarity as we have in our model, and hence the strength of the signal in the iterative stack will be also gradually decreased on iterations.

Therefore, we should carefully choose the number of iterations for the output of our iterative process. In our modelling test, for example, the number of iterations which gave the best display was 5 or 6; and it might be reduced to 3 or 4 in real data processing.

It is also suggested the muting function should be included in the process of the iterative stack. Naess (1979) commented that muting is not required because the stretching effect does not bother the iterative stacking too much. That is only based on his simple modelling result. If we consider the disturbing events from shallow paths and the fact that amplitude changes versus incidence angle, a proper muting will be highly helpful in regulating signal responses especially for shallow depths.

N LEGS: 9.		CLIP:	0.8279
SOURCE: P			
S DEPTH: 0.01		G DEPTH: 0.01	
R: 100.00 DZ: 0.00		DR: 50.00	
VP	VS	DEN	ZI
1.50	0.10	1.00	150.00
2.00	1.11	2.00	250.00
3.00	1.67	2.20	600.00
3.50	1.94	2.50	99.99

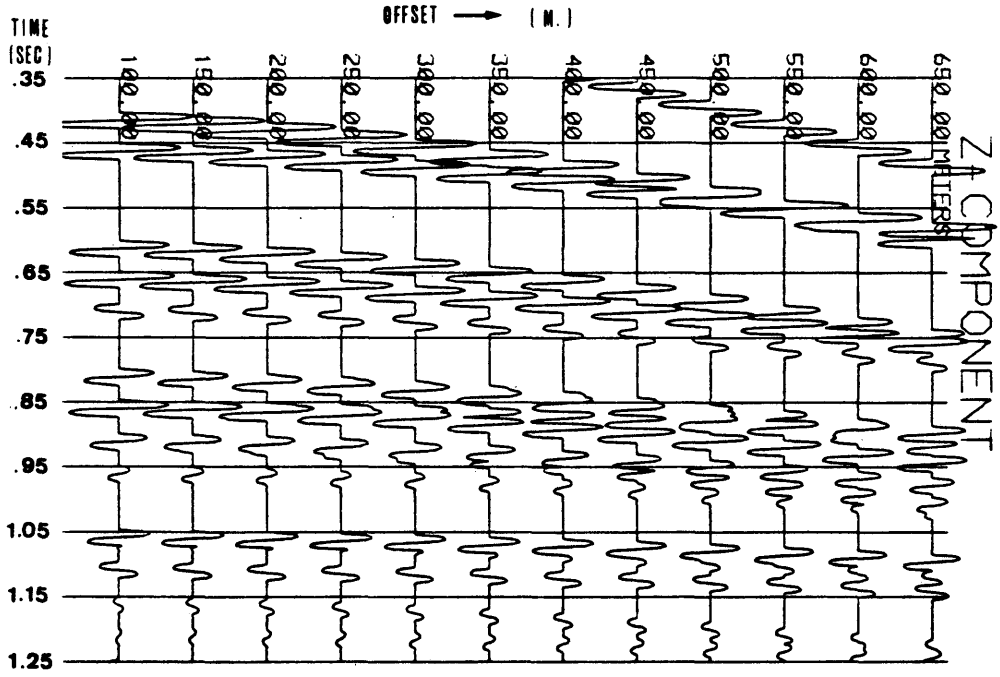


Figure 3-14, 12-fold CDP-gather from the 3-layer model generated by GRAY suite.

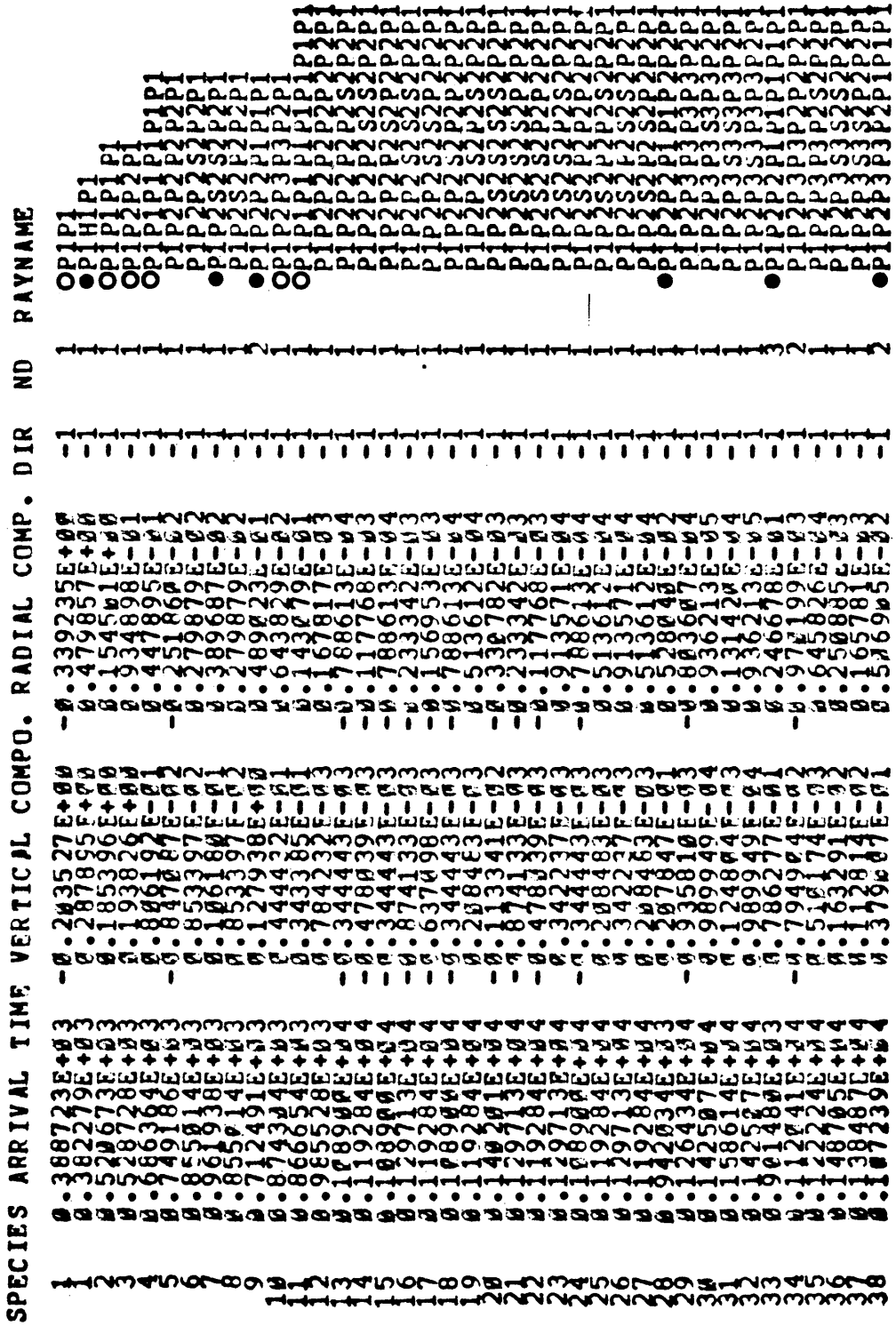


Table 3-1, The rays generated by GRAY suite from the receiver #9 of the 3-layer model. (circle marks those rays appeared in the previous model, black dot indicates the most disturbing events).

4. COMPARISON WITH THE OTHER ALGORITHM

The goal of seismic data processing is to suppress noise and enchance the signal. Random noise is usually reduced by increasing the number of elements such as detectors or sources in the trace mix. Horizontally travelling noise is suppressed by patterns of sources and detectors and by use of frequency and velocity filters. Noise in the reflection path such as ghosts, reverberations and multiples are suppressed by deconvolution or horizontal stacking.

From previous examples, the algorithm of iterative stacking has shown its excellent ability to cancel noise. Now we want to compare its performance with that of other algorithms used in the different categories of noise suppression. We would also like to know whether or not the iterative algorithm is compatible with the other algorithms if they are conducted consecutively.

4.1 Comparison between the Straight, Weighted, and Iterative Stack in Suppressing Random Noise

This comparison is of special interest because straight and weighted stacks are the most common routine

procedures. The modelling method used for this comparison is similar to the one used by Robinson (1970). The model data is a six trace CDP-gather and it is assumed that all traces have already been properly corrected for NMO; the signal consisting of a sequence of evenly-spaced Ricker wavelets. The time extent of windows was short enough for us to ignore the possibility of the attenuation. The signal on each trace differed only in its scale. The noise on each trace was independently generated by convolving a Ricker wavelet with a random series of spikes.

Signal scale could be arbitrarily assigned to each trace and set to 1.0 for the first synthetic trace. S/N energy ratio was computed from the total signal energy and total noise energy throughout the extent of the window. The window time extent (trace length) is 280 samples.

Figures 4-1 to 4-8 show stacking results from different combinations of signal scale and S/N energy ratio. Each figure shows the 6 synthetic CDP-gather, the desired output (true signal), the straight stack, the iterative stack (the 4th-iteration version), the optimum weighted stack, and the iterative stack after a process of optimum weighting.

In figures 4-1 to 4-5, the modelling CDP traces were assigned equal signal scale but varied S/N energy ratio

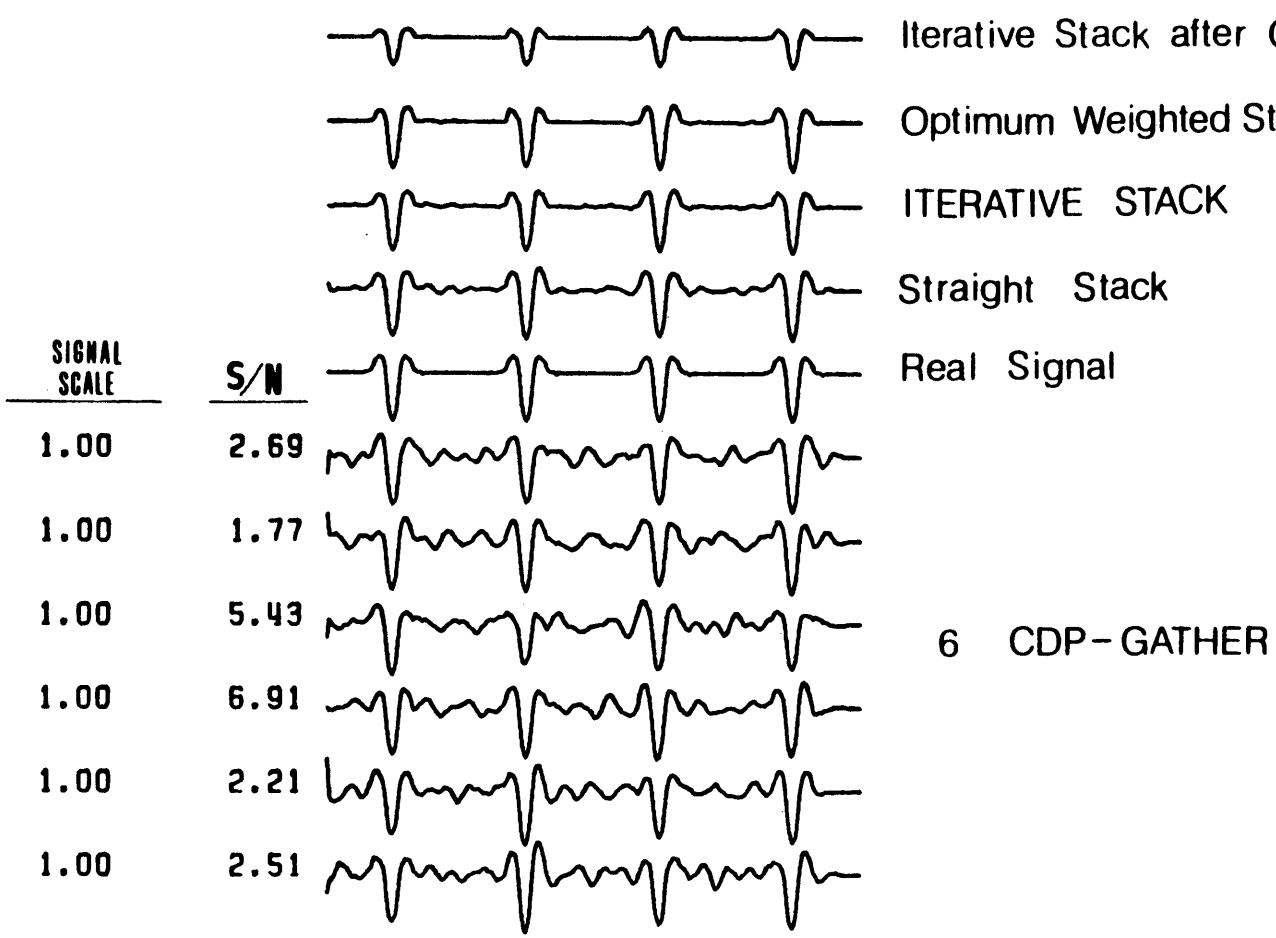


Figure 4-1, Stack from different stacking algorithms with very high S/N ratio.

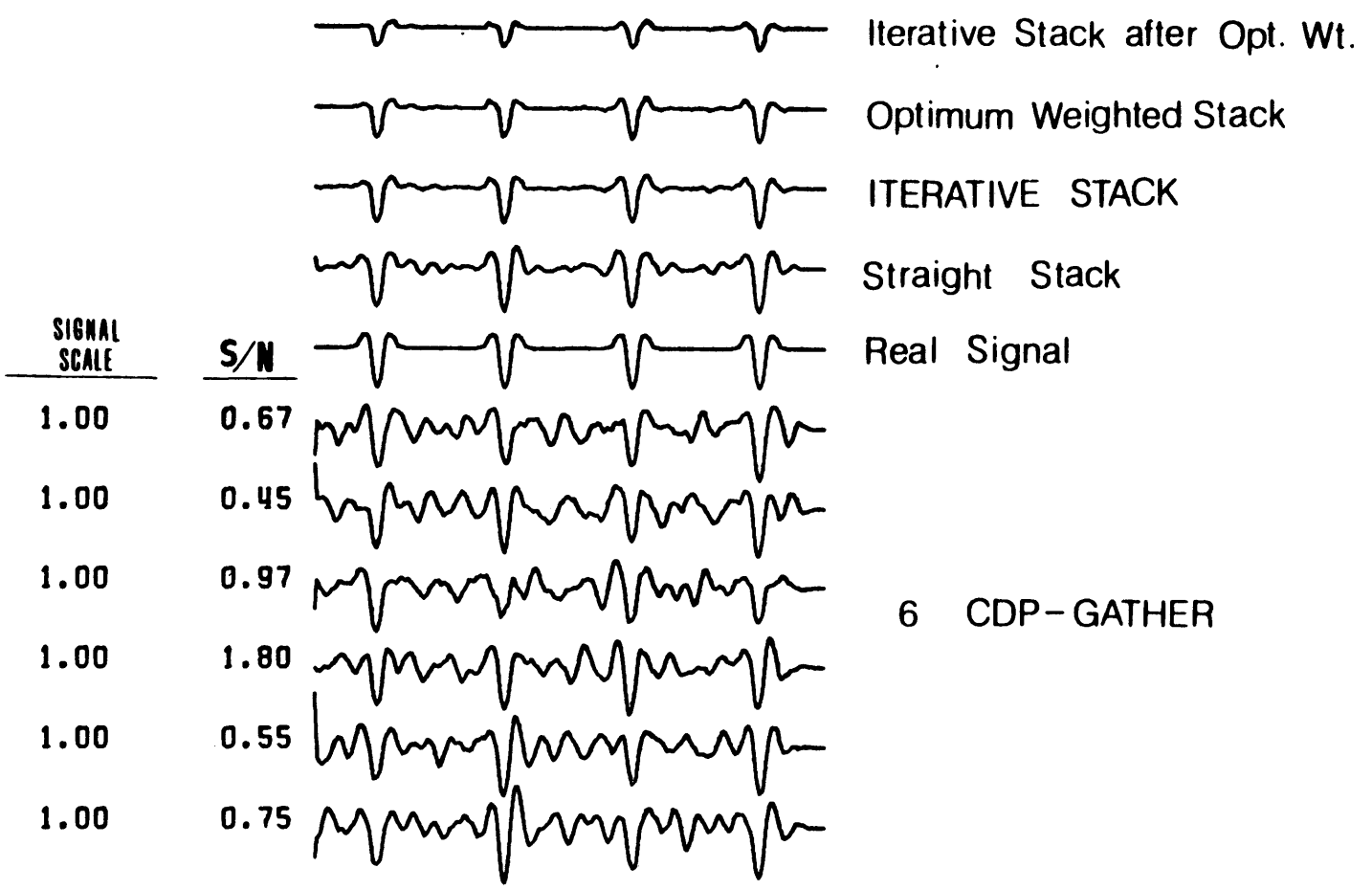


Figure 4-2, Stack from different stacking algorithms with high S/N ratio.

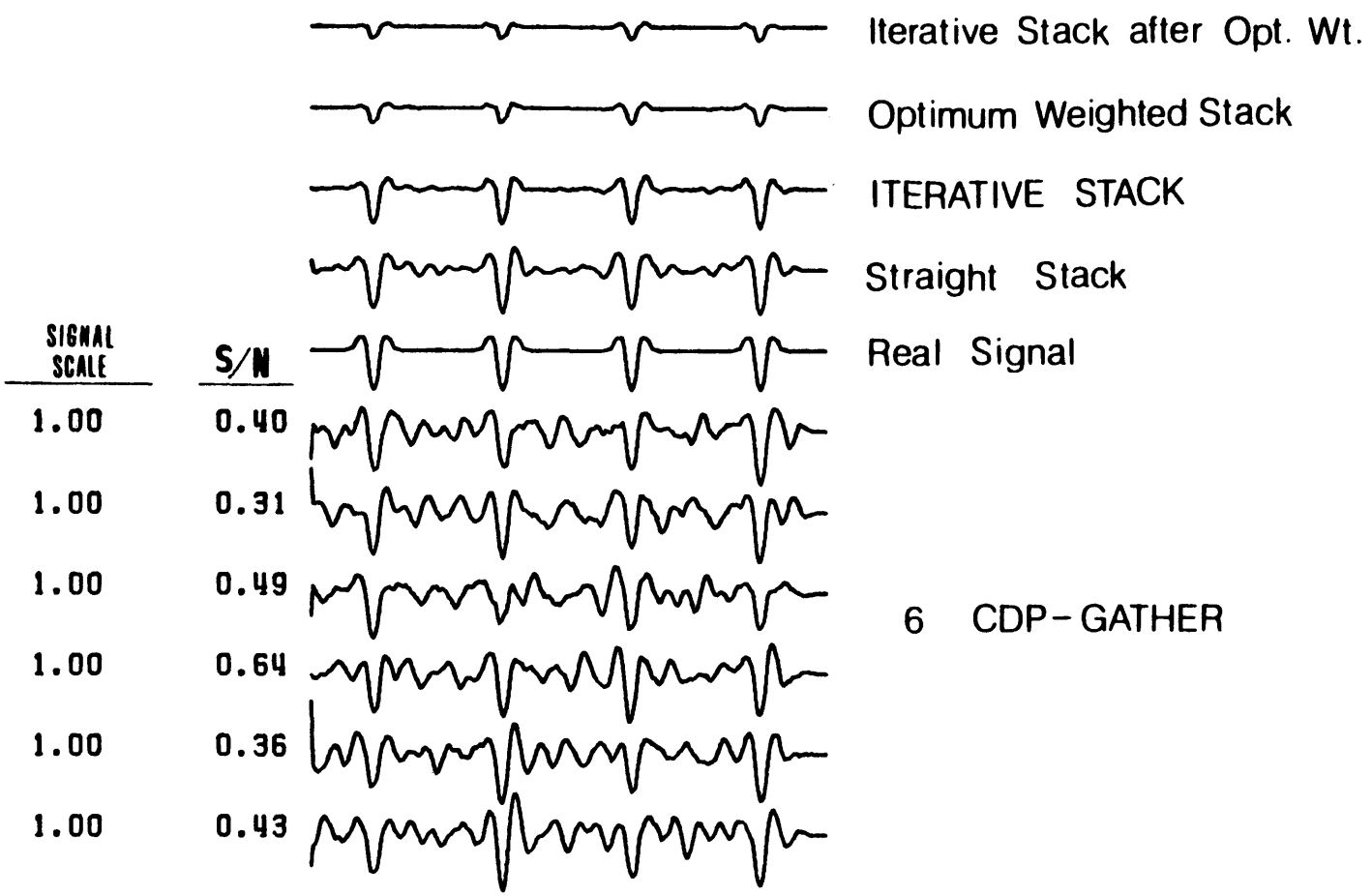


Figure 4-3, Stack from different stacking algorithms with moderate S/N ratio.

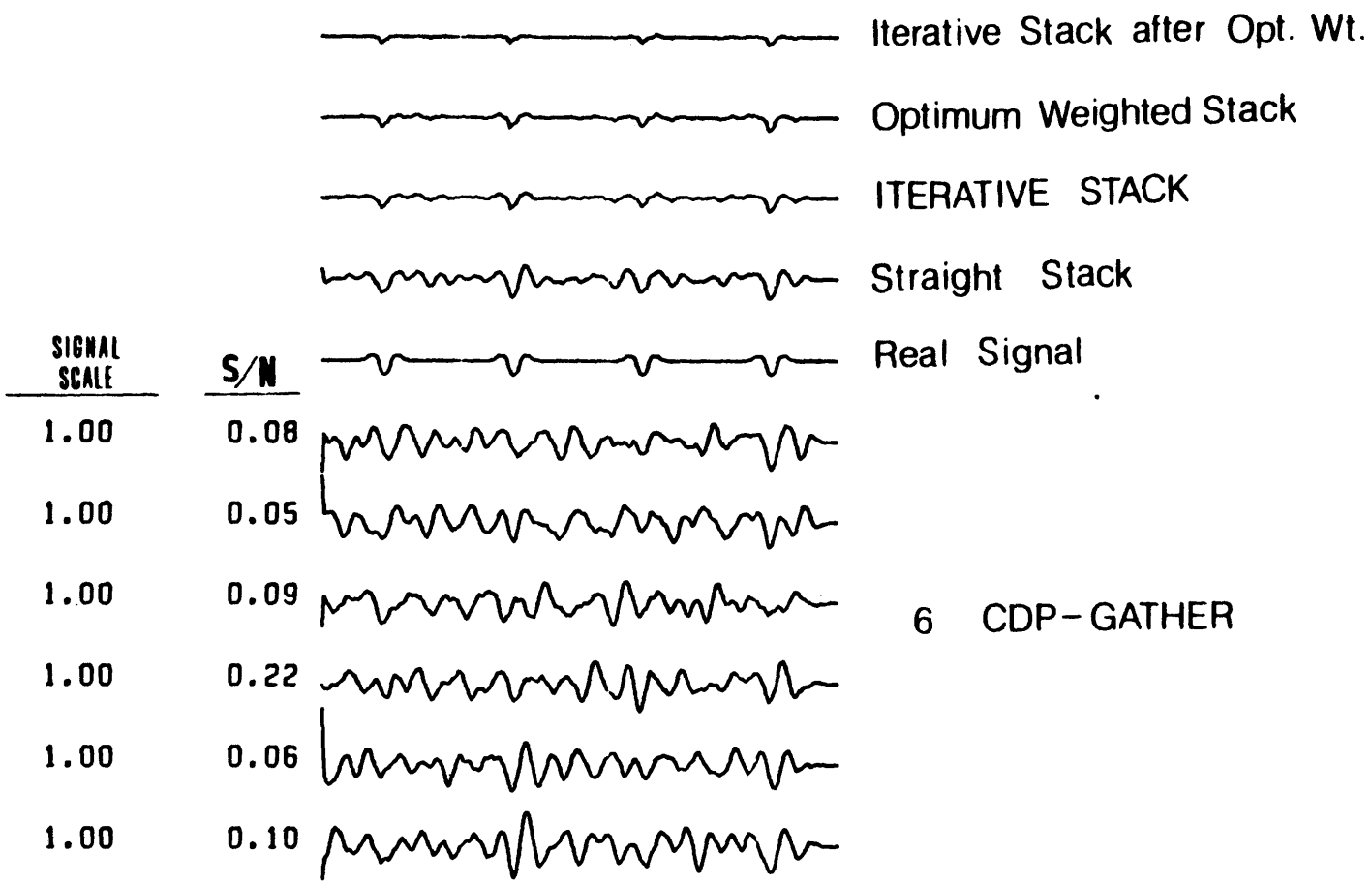


Figure 4-4, Stack from different stacking algorithms with low S/N ratio.

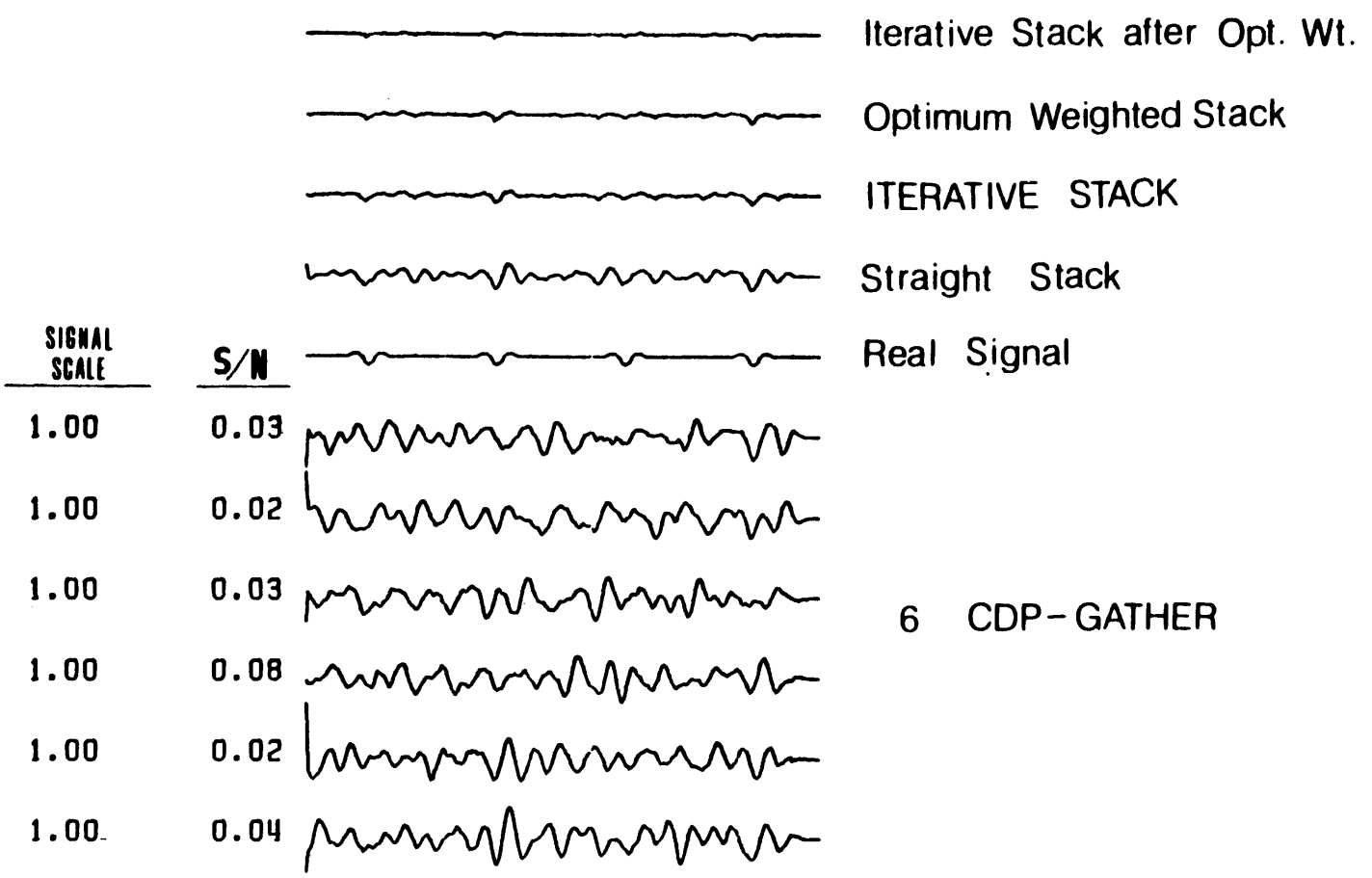


Figure 4-5, Stack from different stacking algorithms with poor S/N ratio.

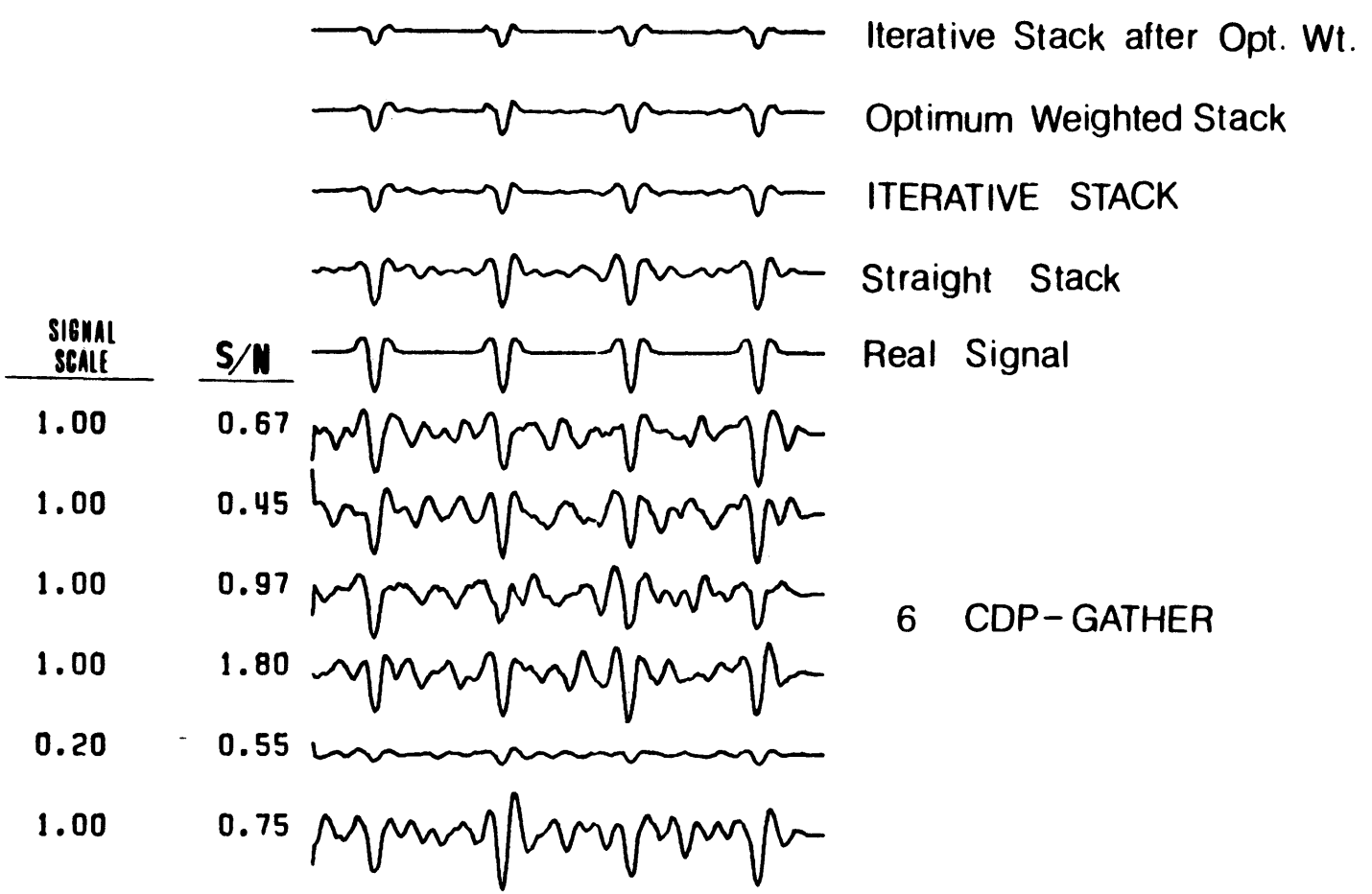


Figure 4-6, Stack from different stacking algorithms with moderate S/N ratio but varied signal scale.

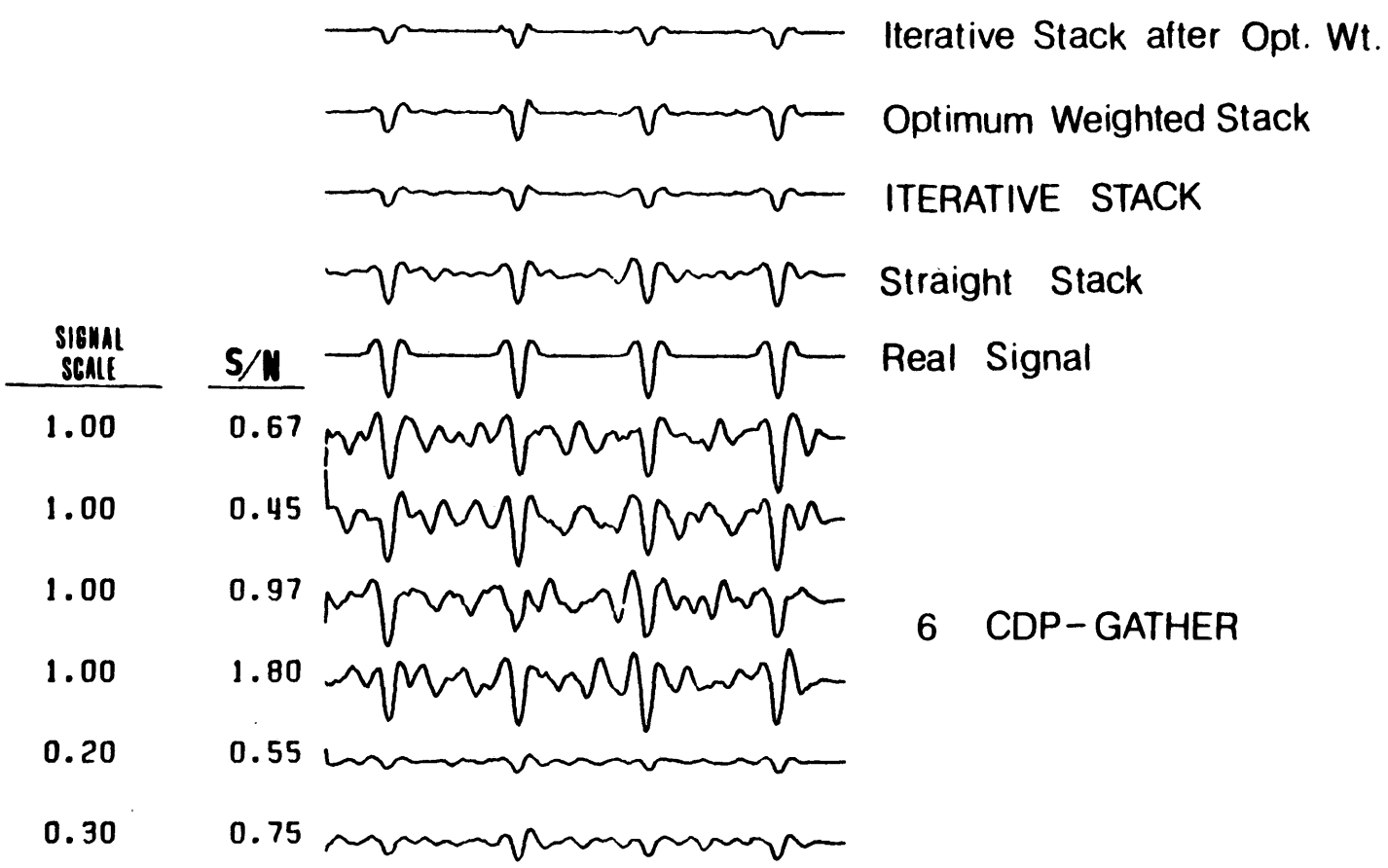


Figure 4-7, Stack from different stacking algorithms with moderate S/N ratio but varied signal scale.

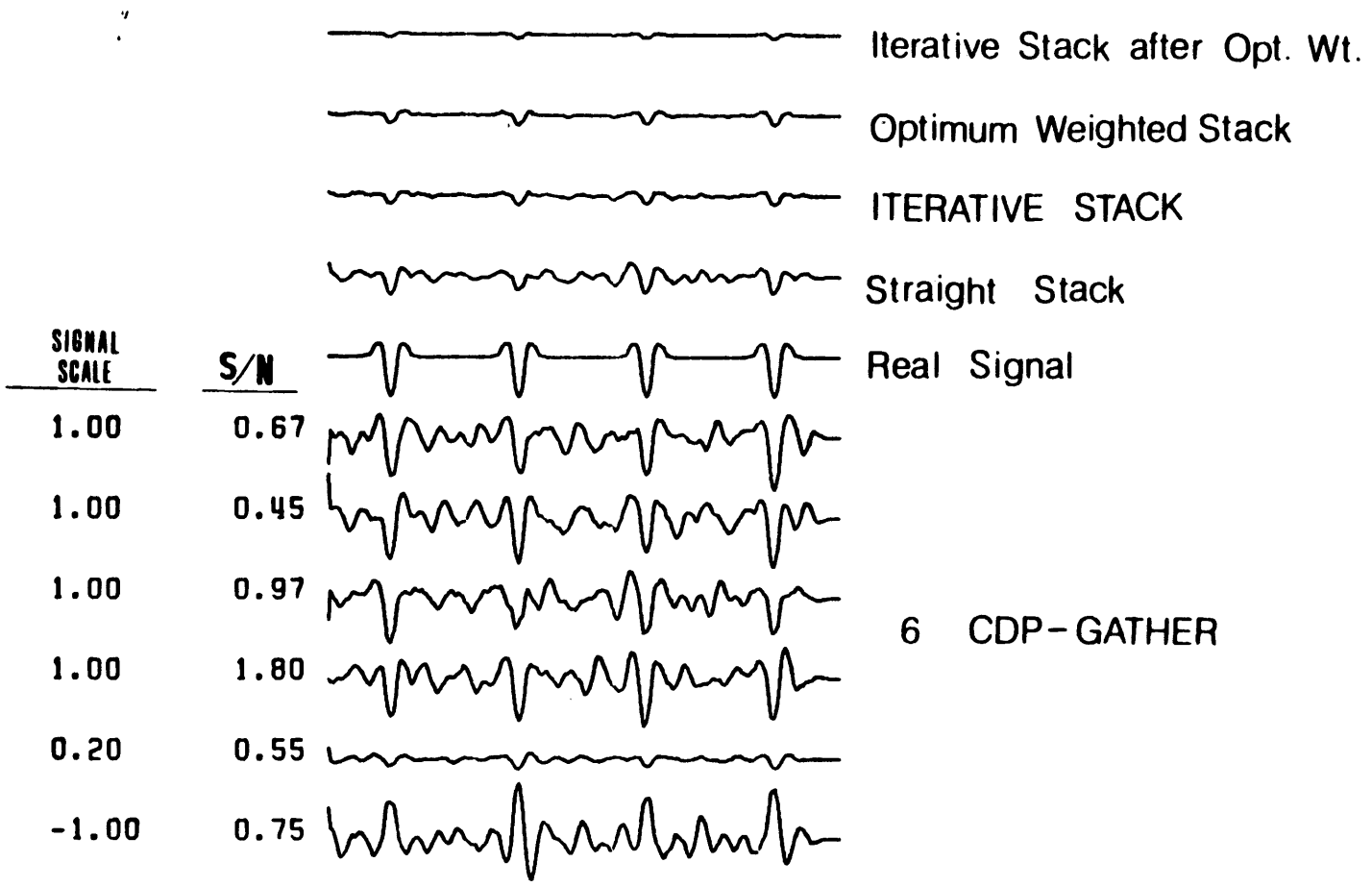


Figure 4-8, Stack from different stacking algorithms with moderate S/N ratio but varied signal scale.

from the highest value(3.5) in figure 4-1 to the lowest value(0.036) in figure 4-5. In figures 4-6 to 4-8, a moderate S/N energy ratio remained constant, while different signal scales were assigned in one or two traces. Therefore, the influence of both the signal scale and S/N ratio on the stacking traces could be observed independently.

The resulting iterative stack is obviously much better than the straight stack in all cases. Hence, the comparison concentrates only on the performance of the optimum weighted stack and iterative stack. My observations could be summarized as follows:

(1) If the signal scale is kept constant among all CDP traces as in the cases in figures 4-1 to 4-5, then:

(1.a) For a high S/N ratio case as in figure 4-1, both optimum weighted stack(OW) and iterative stack(IS) are excellent in enhancing the signal. The iterative stack after optimun weighting(ISOW) shows a smaller output signal than OW or IS.

(1.b) For a moderate S/N ratio as in cases of figures 4-2 and 4-3, IS is more able to emphasize the signal than OW, while ISOW shows no improvement over that of OW or IS and the signal actually gets worse.

(1.c) For a poor S/N ratio as in figures 4-4 and

4-5, IS seems slightly better than OW, and ISOW is still the worst ot them.

(2) If the signal scale varied and the S/N ratio were kept about the same as in figures 4-6, 4-7 and 4-8, the result of IS is not so remarkably superior to that of OW as shown in cases (1.b) above, ISOW is improved and might be as good as IS in the case of fig 4-6. The reason is that a weighting proceduce tends to pull the signal scales back to an equal level on all traces, which benefits the iterative algorithm of ISOW.

(3) It must be noted that the true optimum weighting stack is based on prior knowledge of the synthetic data in this model. In practice this knowledge is hard to obtain. Robinson (1970) has stated that the best statistical approximation to an optimum stack is essentially one-half as effective as the true optimum technique at improving the S/N ratio on a stack over the ordinary stack. The IS process, on the other hand, does not require any prior assumption about the signal scale or S/N ratio.

(4) The process of OW and IS seems incompatible, which was illustrated by the results of ISOW from those figures. It indicates that the OW process should be avoided in routine seismic work if IS will be executed afterward. However, to make a further analysis, the process of OW

actually consists of two functions: one is to give a big weight to the trace of low signal scale; the other is to give a small weight to the trace of high noise ratio, called the diversity stack. Actually the high noise inside the trace does not bother the performance of IS, while the equalizing of signal scale on all traces will benefit IS. Therefore, the optimum weighting sense for IS is just on the equalization of the signal scale. A cross-product among traces (see equation 6) will provide the value of signal scale of each trace and derive a weight to each trace for scale equalizing. This will improve the performance of IS as in the case of figure 4-7 into the better result of figure 4-3.

In practice, the signal scale for all traces can be assumed to have about the same value. In order to improve the performance of iterative stacking, manual editing of bad traces will still be valuable, whereas diversity stacking should be eliminated from the routine processing sequence.

4.2 Comparison between the Iterative Stack and the Velocity Filter in Suppressing Surface Travelling Noise and Multiples

The objective of this section is to compare the performance of iterative stacking and velocity filtering in suppressing surface travelling noise as well as multiples. Since the velocity filter is commonly used to improve the data continuity prior to the stack, we would also like to know the compatibility of the two processes.

Velocity filters are two dimensional linear operators that can extract signals from a background of coherent noise. These filters may be designed to pass only events with a particular apparent velocity. A typical example of a display of seismic signal and noise spectra in the f-k domain is given in figure 4-9.

Apparent velocity in f-k domain is represented by dip, which is defined as Δt per trace. Noise that consists of responses from surface travelling waves will have a slower apparent velocity; a signal that is reflected at nearly vertical incidence and received by geophones at various distances almost at the same time, will have a much higher apparent velocity (SEG 1974). Therefore, noise which overlaps the signal in x-t space could easily be separated

and filtered in the f-k domain. Also primary reflections and any overlapping multiples usually exhibit different time moveouts which can allow us to selectively discriminate against multiples by using f-k processing on NMO-corrected CDP-gathers (Sengbush 1983). The velocity filter may also be oriented to fit particular dips in the final seismic section to enhance geologic structure (SEG 1974).

The model data used here is generated from the geologic model of figure 3-1. It consists of one waterbottom and two primary reflections and five multiples confined in the water layer (long-period multiples). An extra horizontally travelling event is added for comparative purposes. Figure 4-10 shows the generated CDP-gather after NMO correction. Assuming that NMO corrections have been adequately applied, the primaries are flat. Hence, a velocity filter could easily be designed to eliminate all but the flatest noise components from the CDP-gather.

The f-k filtering is accomplished by zeroing out the part or f-k space in which the multiples and noise reside. We will zero out at least half of the spectrum to suppress the multiples. Figure 4-11 shows the zeroing-out area in f-k domain and figure 4-12 shows the data transformation

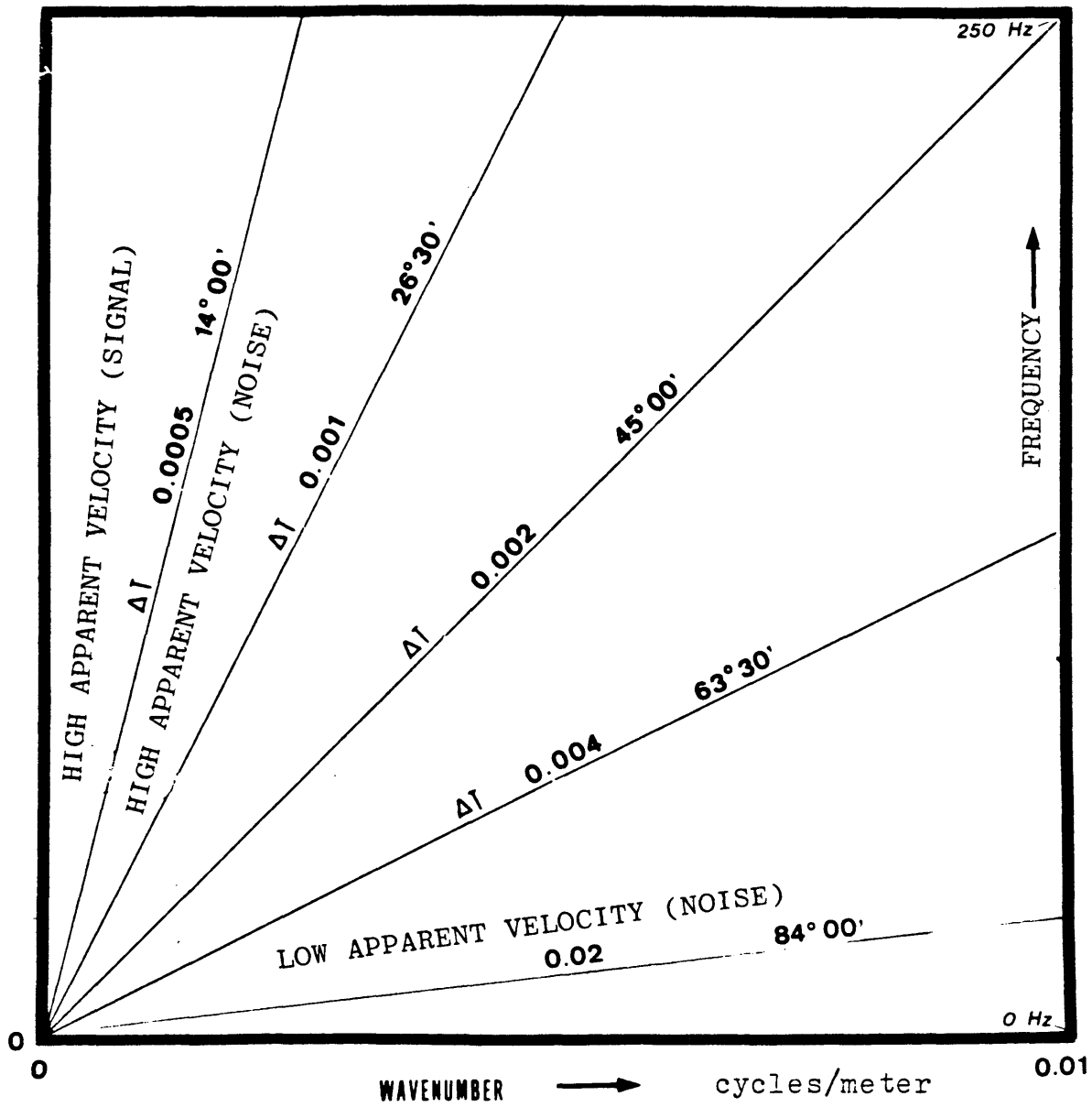


Figure 4-9, Stepout/bandpass relation in f-k domain for CDP-gather traces with sampling rate of 2 ms.

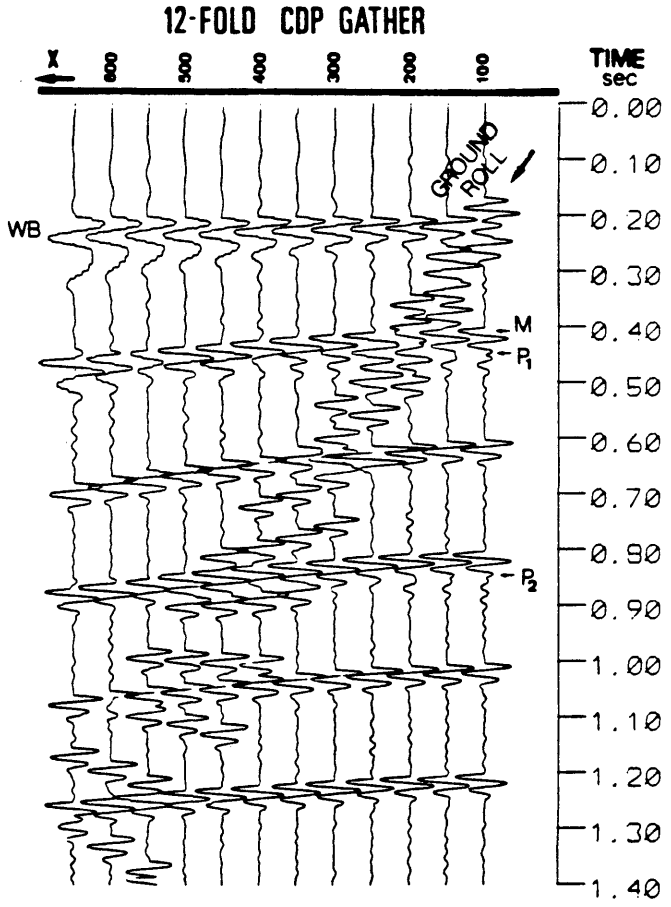


Figure 4-10, Modelling traces generated from figure 3-1 with horizontally travelling noise added, already NMO corrected.

back into x-t space, where surface waves have been successfully eliminated but multiples still exist and seem to get better alignment in the horizontal direction. To compare it with the performance of the iterative stack, we use a CDP-gather of 5-iteration traces from the same model. They are put together in figure 4-12 in order to recognize how significantly the results from these two processings differ.

Another f-k filtering test is made by zeroing below the 45 degree line ($\Delta t=0.002$ sec) in the f-k domain. The masked area includes the surface travelling wave. The resulting x-t plot versus the plot of a simple iterative stack are also illustrated in figure 4-13.

The comparison from these figures show that both methods can delete the surface wave noise, but the iterative stack can totally clear out multiples, while the velocity filter cannot. Even in a model of 36 CDP-gather traces (Sengbush 1983) having better conditions for the f-k filtering process because of more horizontal sampling, was unable to remove multiples in near traces using velocity filtering.

There is no doubt that the iterative stack is more effective than the velocity filter in suppressing low velocity noise, especially for multiples. If that is the

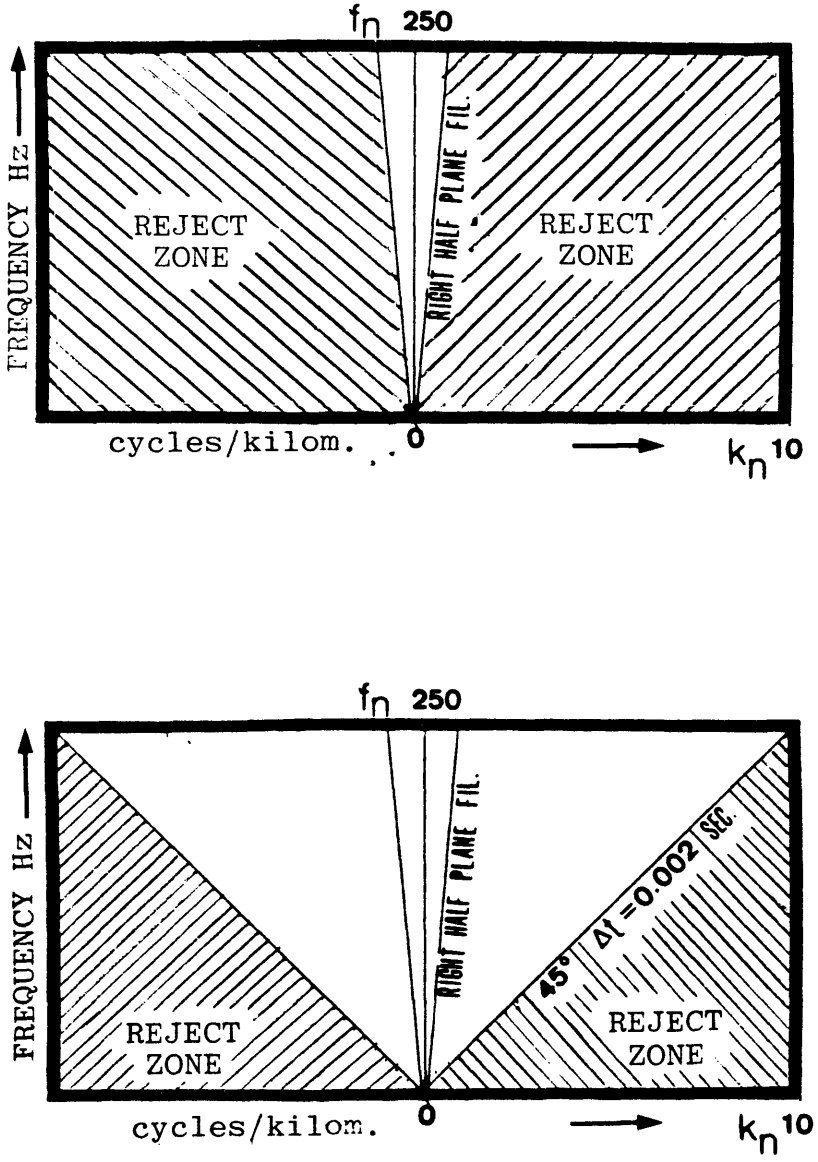


Figure 4-11, Reject zone in f-k domain, upper: Mask for figure 4-12; lower: Mask for figure 4-13.

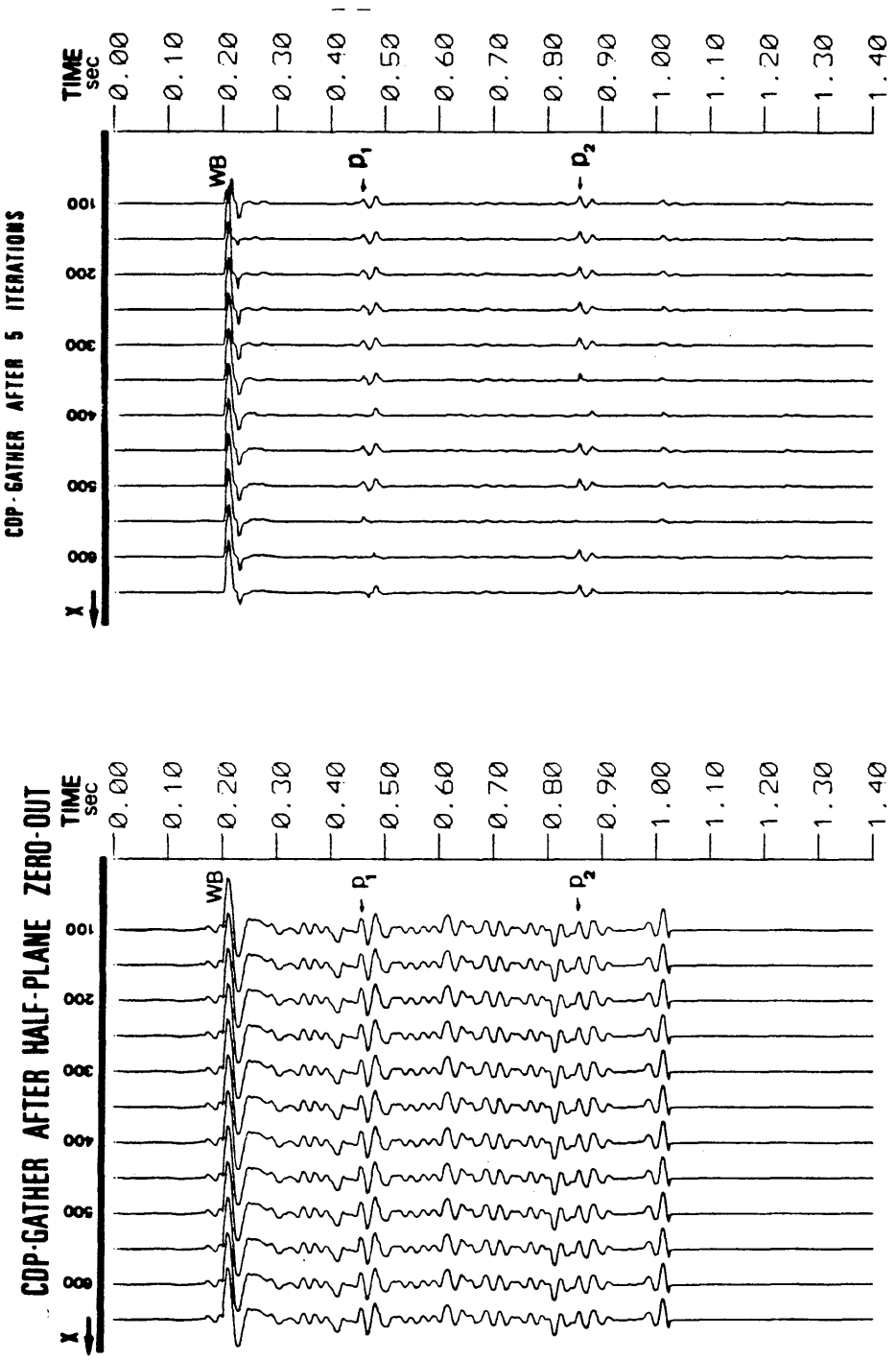


Figure 4-12, Comparison of the f-k filtering result by the half-plane zero-out (left) and the 5-iteration output of the iterative stack (right).

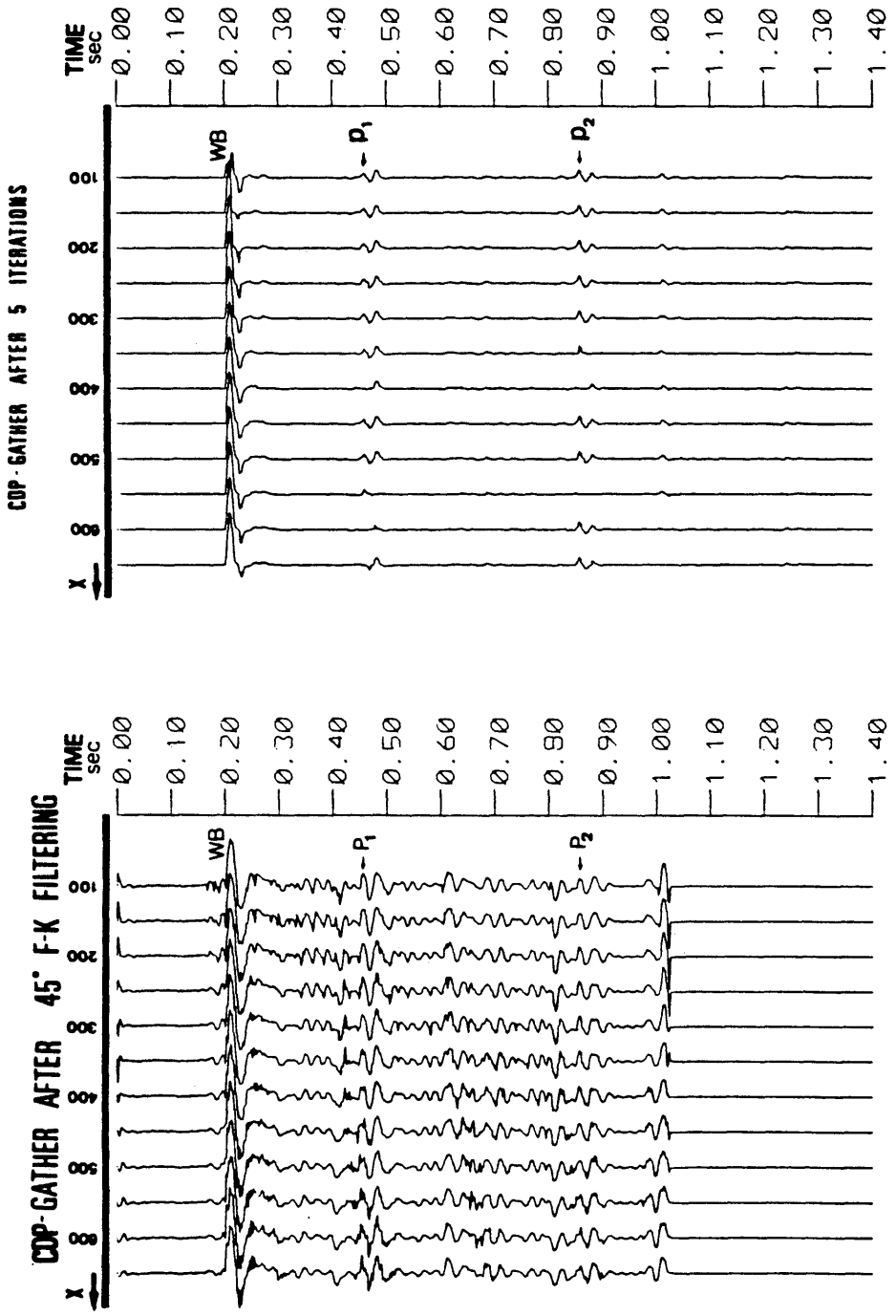


Figure 4-13, Comparison of the f-k filtering result by 45 degree zero-out (left) and the 5-iteration output of the iterative stack.

case, should we eliminate the velocity filtering job in routine seismic work and replace it with iterative stacking? In the conventional routine processing it is usually believed that multiples may be suppressed first in the CDP-gathers by a velocity filter, and then suppressed further by subsequent stacking; if that is true and if iterative stack is subsequently conducted instead of straight stacking, we would like to know if the result is constructive or destructive.

We are reminded that all 2-dimensional filters can be seriously affected by a problem of aliasing in the horizontal direction; signal frequencies which are greater than one half the sampling frequency are folded back into the spectrum as low frequencies equal to the difference between the sampling frequency and the signal frequency. The best solution to avoid aliasing is simply to sample more otten horizontally, which means shortening the spacing between receivers. The maximum spacing for proper recording in a given area must be one half of the slowest horizontal velocity divided by the highest signal frequency (SEG 1974).

In our model, the maximum spacing should be less than 20 meters according to that standard, whereas the spacing we used is 50 meters. Thus, aliasing exists and the

frequency content of our signal has been damaged after the 2-dimensional transformations back in the x-t space as in figures 4-12 and 4-13. The iterative stack was applied to these filtered gathers and the result displayed in figure 4-14 (from gather after right-half plane zeroing) and figure 4-15 (from gathers after 45 degree slicing). In the same figures the iterative stack processed directly from original data is also illustrated for comparison. It is evident that iterative stacking after velocity filtering is not able to eliminate multiples as it does individually. Even in higher iterative versions, the multiples still remain unaffected.

Accordingly, we suggest that velocity filtering should be discarded from pre-stacking procedures because its effect is marginal compared to iterative stacking, and it will damage the subsequent iterative stack especially when an aliasing problem exists, which it usually does.

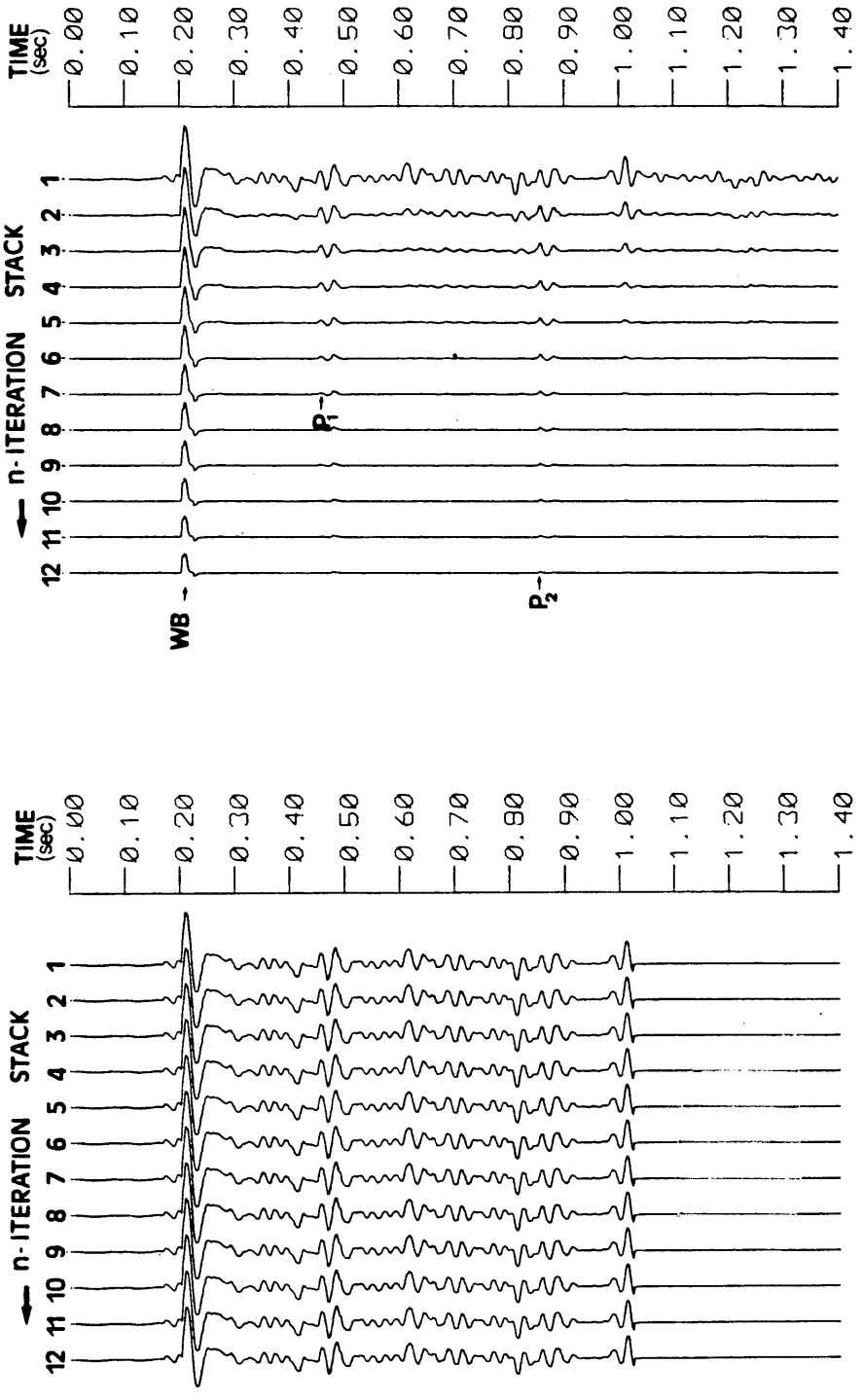


Figure 4-14, Left shows the result of iterative process applied to the output of f-k filtering (half plane zero-out), right picture is the result of direct iterative stack.

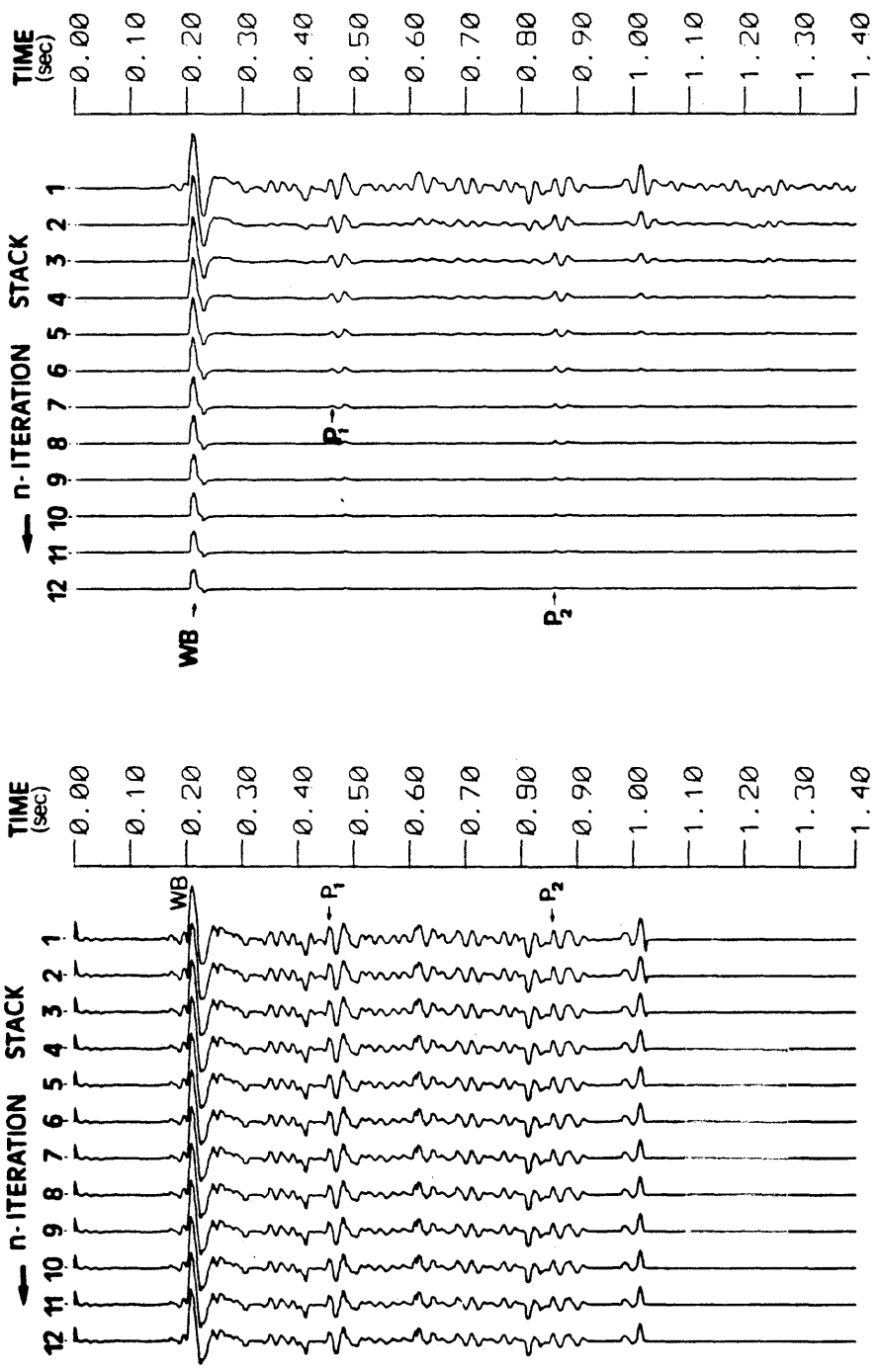


Figure 4-15, Left shows the result of iterative process applied to the output of f-k filtering (45 degree zero-out), right picture is the result of direct iterative stack.

4.3 Comparison between the Iterative Stack and Wiener Deconvolution in Suppressing Periodic Noise

Wiener least-squares inverse filtering (Rice 1962) is one of the most effective tools for the digital reduction or seismic traces, it constitutes the keystone of many current deconvolution methods. The predictive error filtering process, for example, is one of the most flexible deconvolution methods based on that algorithm and, will be used here for our comparison according to the description in papers by Robinson (1966), Robinson & Treitel (1967), and Peacock & Treitel (1969). The Wiener filter is used to deconvolve a reverberating pulse train into an approximation of a zero-delay unit impluse. Generally, it will remove repetitive events having specified periodicities such as ghosts and water-layer reverberations, and occasionally multiples.

The objective of this section is to compare the performance of iterative stack with Wiener deconvolution in suppressing periodic noise. Since deconvolution is usually executed several times in the whole processing procedure before or after the ordinary CDP stacking, we would also like to know the influence of the iterative stack on the Wiener process if iterative stacking is substituted for the

ordinary straight stack.

In order to make a comparative check, we designed a model by adding a ghost reflection to the modelling data of figure 3-1, in whicn the water layer confined multiples could be regarded as a long-period repetitive event, while the ghost is regarded as the short-period repetitive event. The distortion operator involving the ghost is defined as $S(t)+RS(t-\Delta T)$, where the ghost reflection coefficient R is assumed to be -0.95 and ghost delay time ΔT is 40 ms. Figure 4-16 shows a CDP-gather of 12 traces generated from this model.

Two approaches have been used to attack the ghost and multiple in our modelling traces. One approach is to make the iterative stack first and then apply the Wiener process. The other approach is to do the Wiener process first followed by the iterative stack.

In the first approach, iterative stacking was performed first. Figure 4-17 shows the stacks resulting from the iterative process. Random noise and long-period multiples are totally removed from the final stack after several iterations, but ghost reflections remain in the traces because their velocity is so close to the primaries that their response can not be selectively suppressed. It is important to note that the ghosts of waterbottom layer

have not been exterminated but have been distorted due to the stretching effect ot NMO correction. This will distort the subsequent design of the Wiener Operator.

To attack the ghosts in figure 4-17, a Wiener Operator was designed from each trace with a window of 600 samples and filter length or 45 samples. The program to solve the linear equations was obtained from the LINPACK software (Dongara, et. al., 1979), which is in the LBY: library of the DEC 10 mainframe. The calculated set of filters was convolved with each trace to generate an approximate spike, and then convolved with the 40-Hz weighted sine wavelet to give us the final result as shown in figure 4-18.

The result of the Wiener process shown in figure 4-18 is not satisfactory since the ghost is only slightly attenuated. There are three reasons that probably explain this poor performance. One is that only three reflective events are left in the final stacked traces, which is insufficient to emphasize the periodicity. The other is that ghost responses are distorted by NMO stretching. Third, the wavelet is not minimum phase (example of wavelet spectrum analysis is shown in Appendix A, the program 'TRPLOT').

Since the distorted response of the first reflective

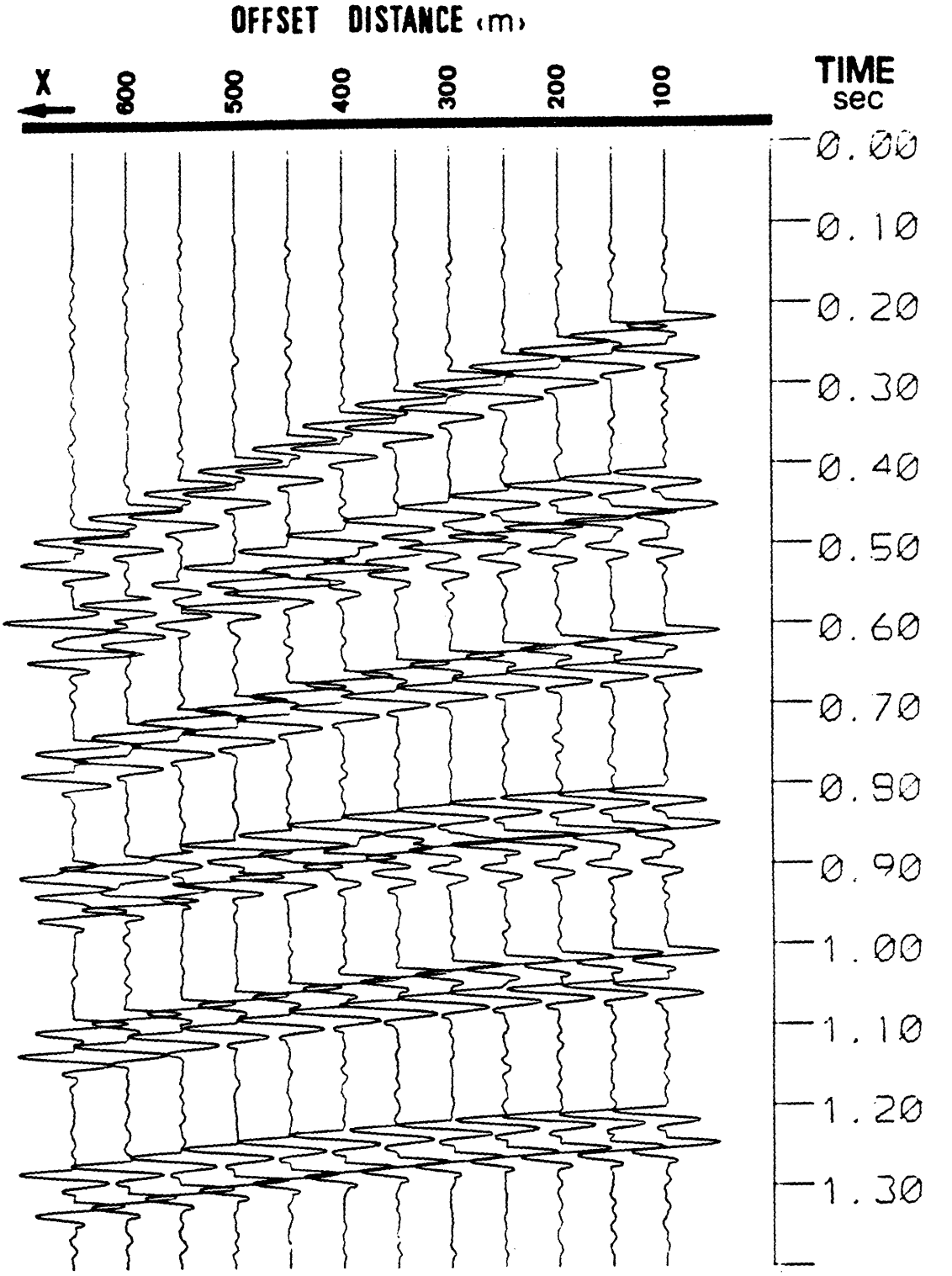


Figure 4-16, 12-fold CDP gather generated from the model in fig. 3-1 with ghost added.

ghost might be the key reason for the poor performance, an attempt was made to develop Wiener filters from the response of the second layer. The result of this Wiener process is illustrated in figure 4-19, which shows some improvement in suppressing ghosts in the second layer but the first layer response was damaged at the same time.

My next attempt to improve the performance of the Wiener process in attacking ghosts on the final iterative stack traces involves the muting skill. Muting is generally performed in the ordinary stacking of CDP-gather. In the early part or the record the long offset traces may be excluded from the stack because they are dominated by refraction arrivals or because their frequency content after NMO correction is appreciably lower than normal traces (Sheriff, 1982) or because the reflection amplitude changes with respect to the incidence angle.

The muting formula I used for that purpose is to exclude data which has NMO time correction over 50 ms. Figure 4-20 shows the resulting stacked traces. Comparing it with the non-muted version of figure 4-17, you will see that the three ghost responses are all sustained in the muted version, and furthermore, the second primary reflection shows greater strength.

The need for the recovery of the waterbottom ghost

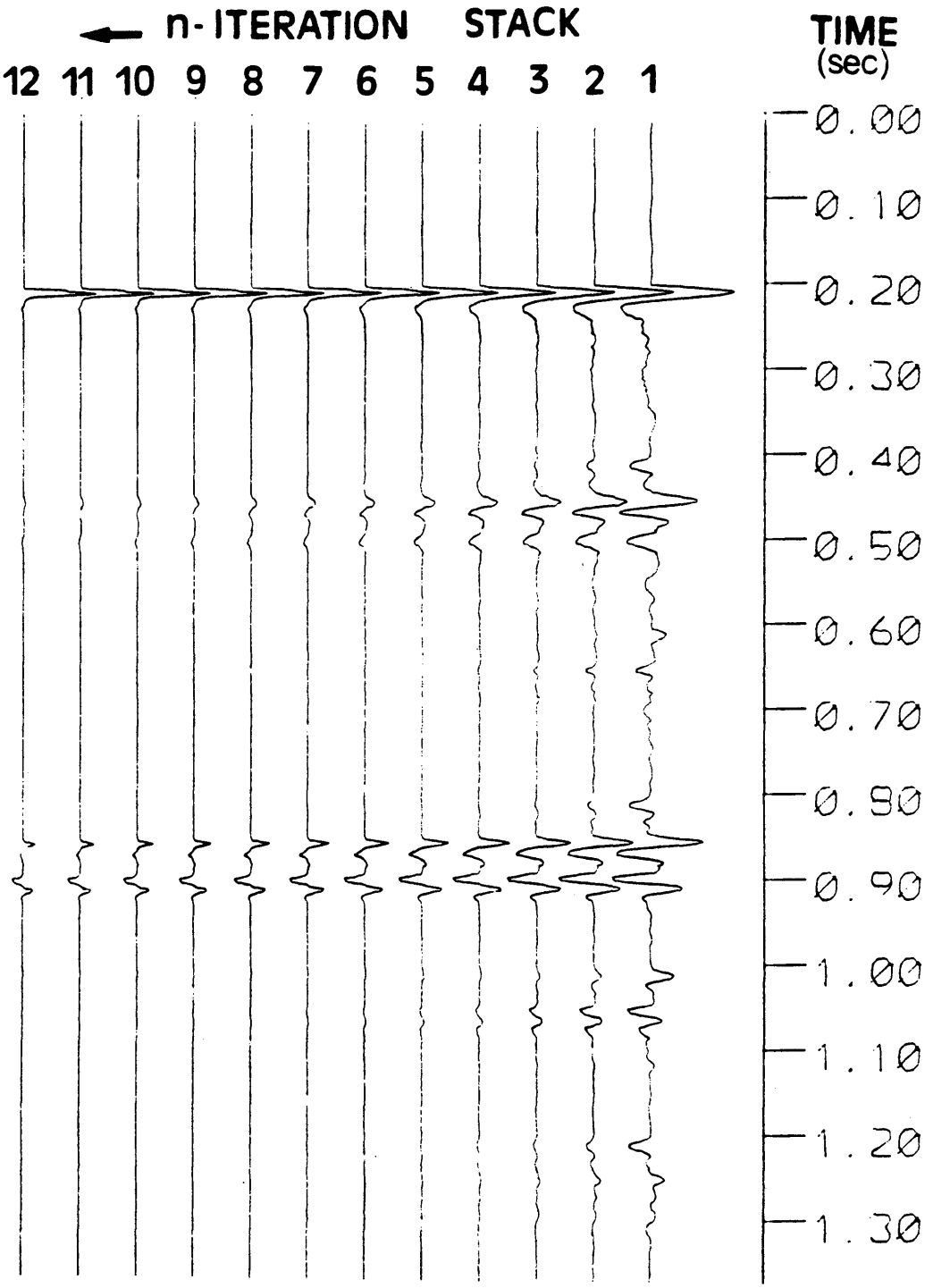


Figure 4-17, Stacks by iterative process on the modelling CDP-gather of figure 4-16.

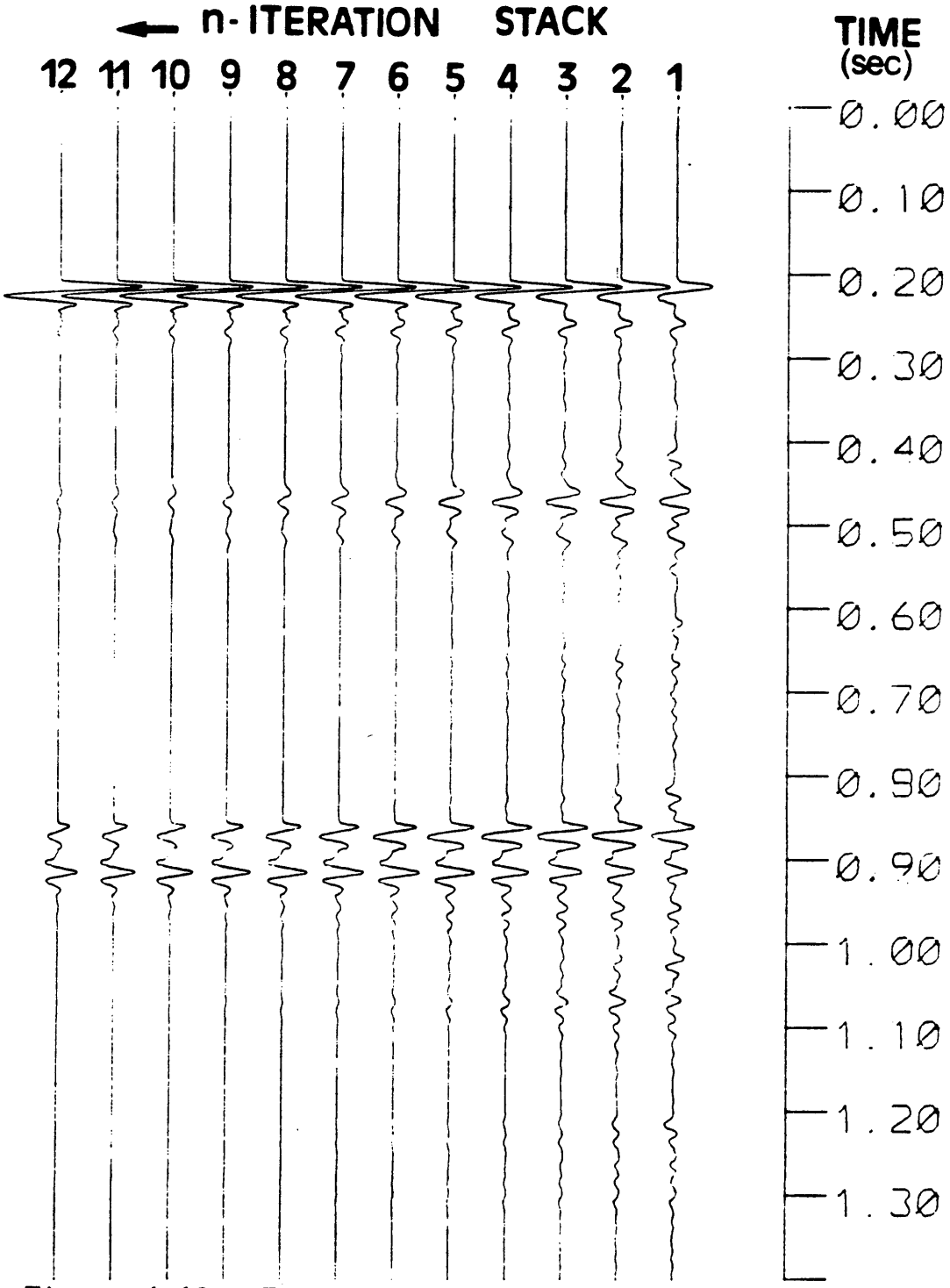


Figure 4-18, The result of Wiener process on figure 4-17.

lies in the fact that this ghost is needed for the design of the deghosting Wiener filter. We performed the Wiener process on this muted version and got the result as shown in figure 4-21. Unfortunately, the ghost, especially at the third layer, was still not suppressed as we had hoped.

It was then guessed that the irregularity in waveform resulting from iterative processing might be the reason that caused the Wiener process to fail, hence, another try was made by using single-trace-processing instead of all-trace-summing for the output of the iterative process, because single-trace-processing has been proven to be better in keeping its original waveform.

Figure 4-22 shows final iterative stacks made by single-trace-processing plus a muting function. Figure 4-23 shows the result after the Wiener process, which is still unable to make any significant improvement in suppressing ghosts. Finally, we tried to use a predictive deconvolution for this job. A predictive error filter was developed from each trace by using an exact ghost-delay time interval of 40 ms. The result of predictive deconvolution is shown in figure 4-24, in which the third layer ghost is still unaffected. If the third layer ghost keeps its original ghost delay, the predictive error process should definitely be effective in removeing it.

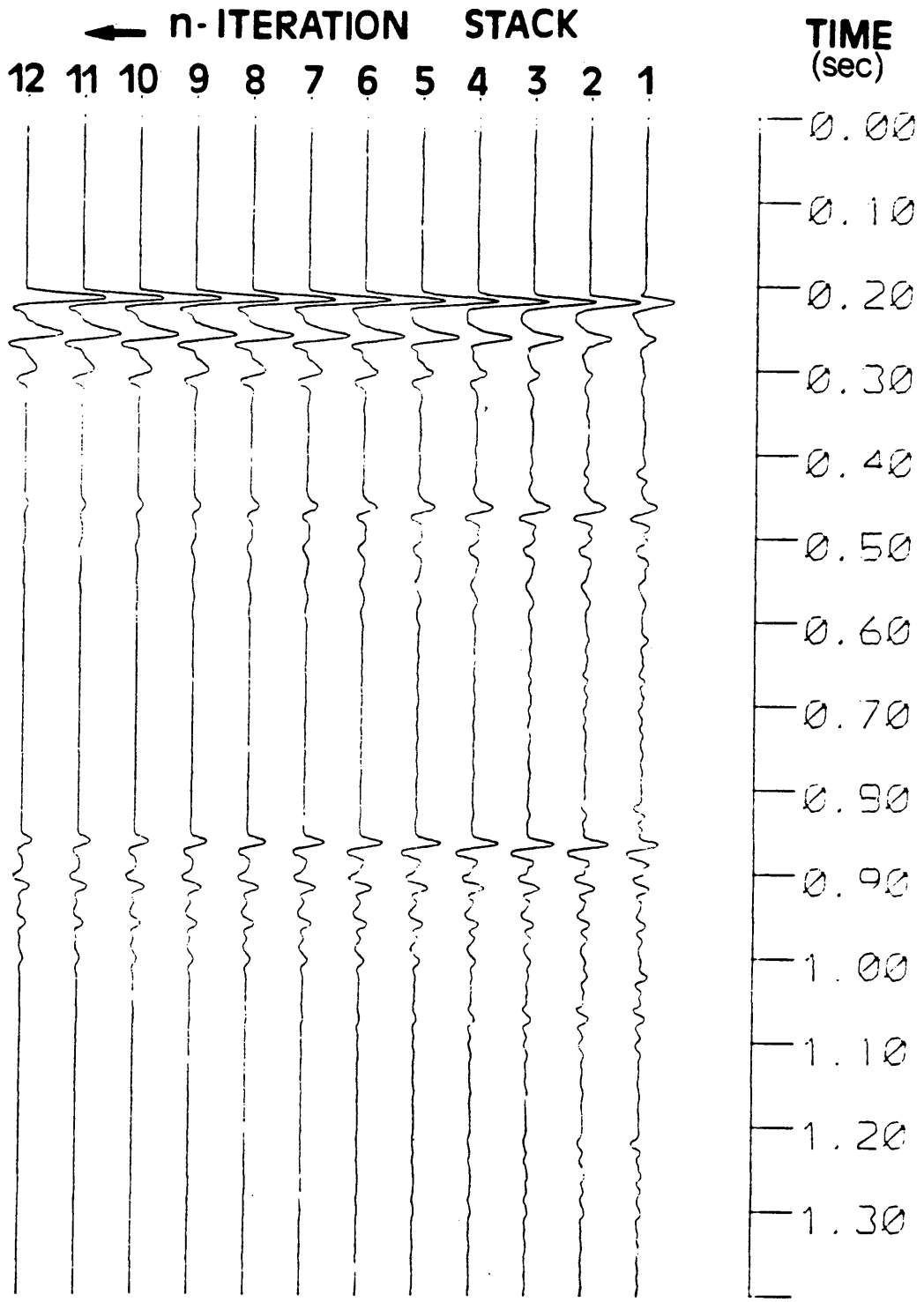


Figure 4-19, The result of Wiener process on figure 4-17 with Wiener operator designed against the 2nd reflection.

Therefore, it is believed that the form of the third layer ghost has been seriously altered after the iterative process because a strong long-period multiple was overlapping it in the original CDP-gather.

If figure 4-24 is the best final stack we can get, we could assume that the normal result of the Wiener deconvolution on the iterative stack traces will be, at most, as good as that shown in the first two layers. However, abnormal results could be seen very often, as happened at the third layer in our test. The conclusion from all these efforts is that the Wiener process could not easily achieve its goal by this approach.

Naess (1979) has stated that "with iterative stack, deconvolution might be better applied between stacks instead of before stack. A few stacks should clean up the traces in the gather considerably and thereby lead to improved design of decon-operators". Our modelling tests just disproved this optimistic assumption.

The second approach, on the other hand, which executes the Wiener process on the CDP-gather traces prior to stacking, has no such problem. First we use the same design parameters to develop the Wiener operator for each original trace, and convolve the trace with the filter to get a de-ghosted trace as shown on figure 4-25. We then

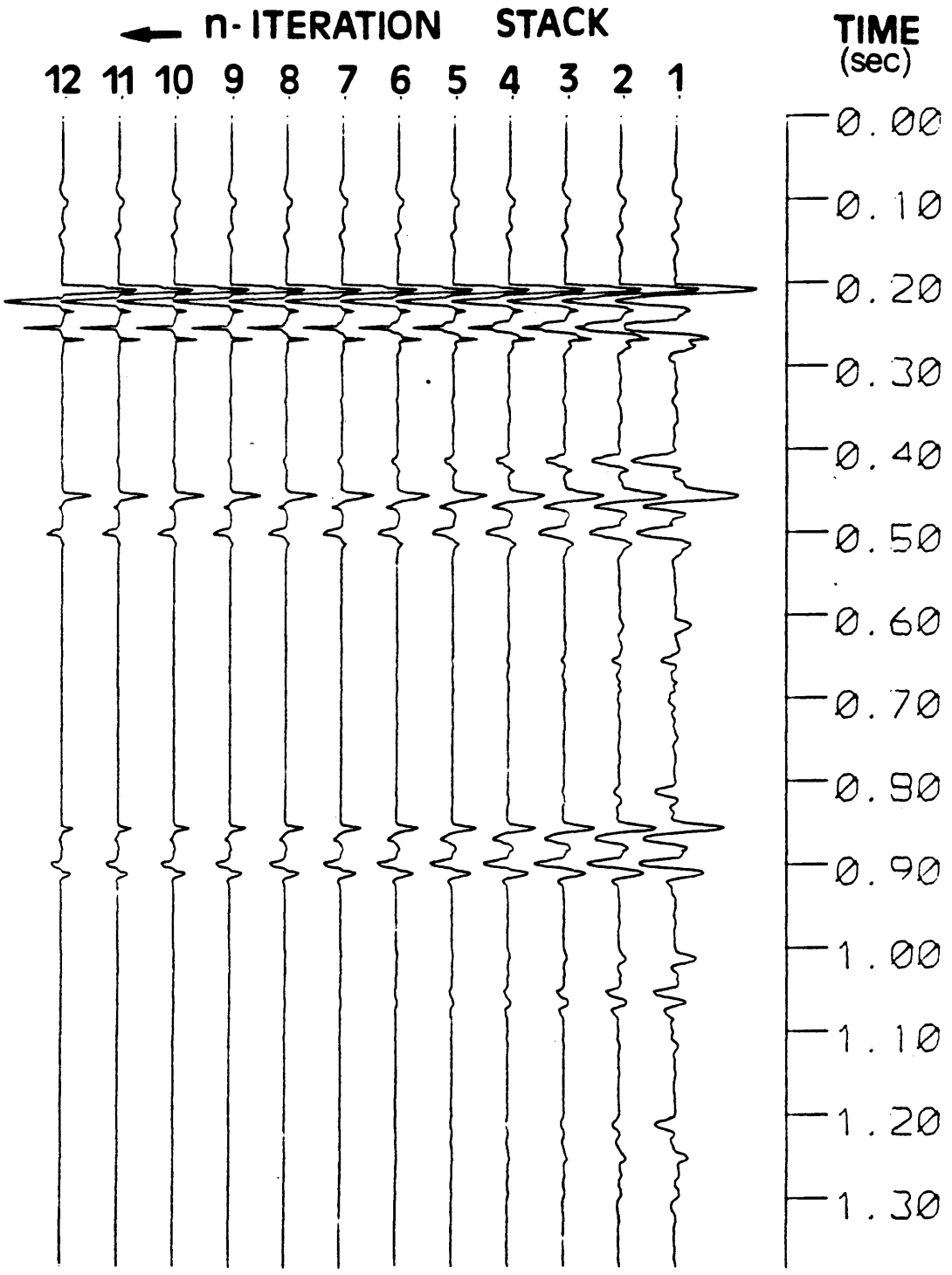


Figure 4-20, Iteration after muting.

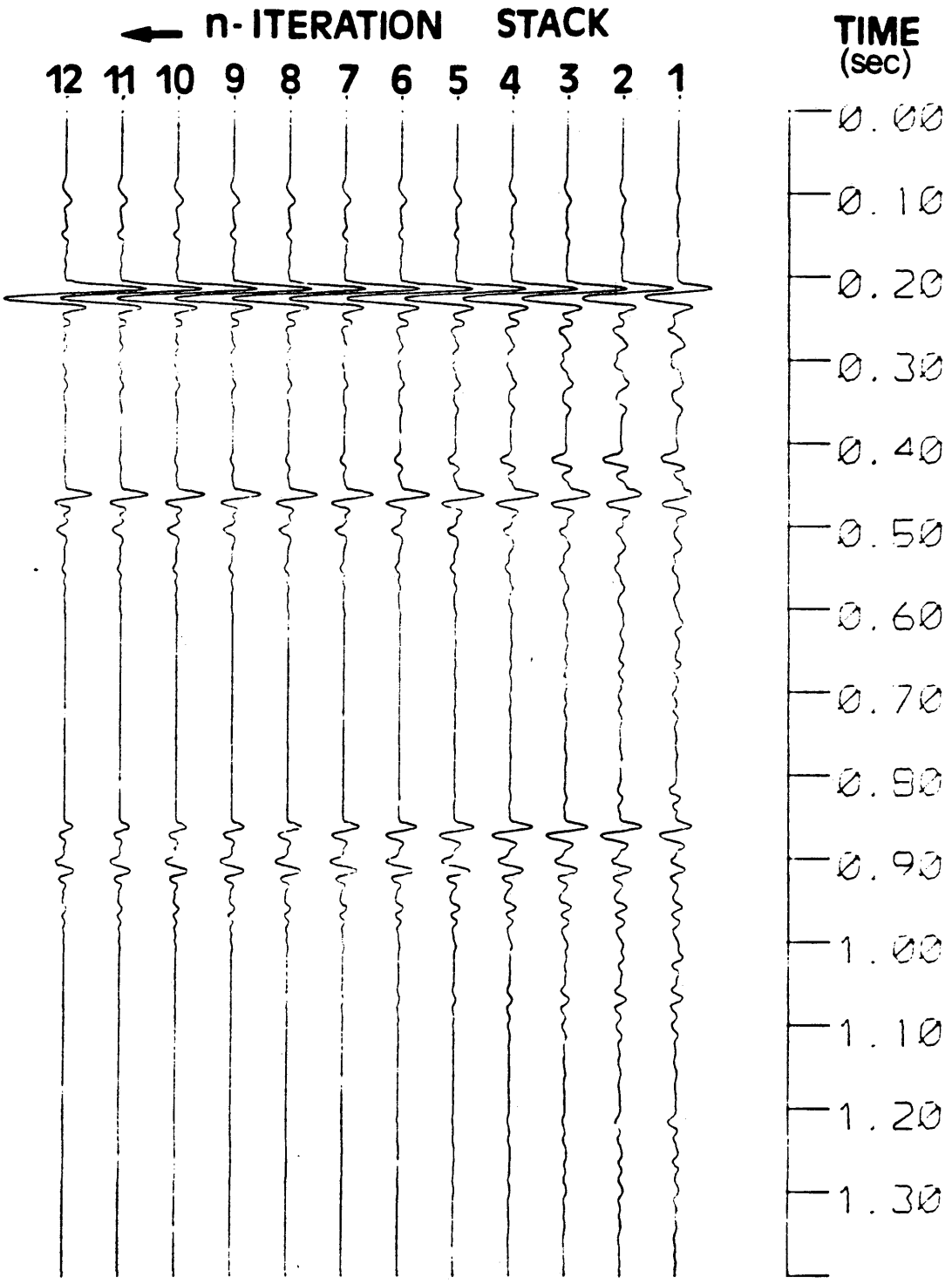


Figure 4-21, The result of Wiener process on figure 4-20.

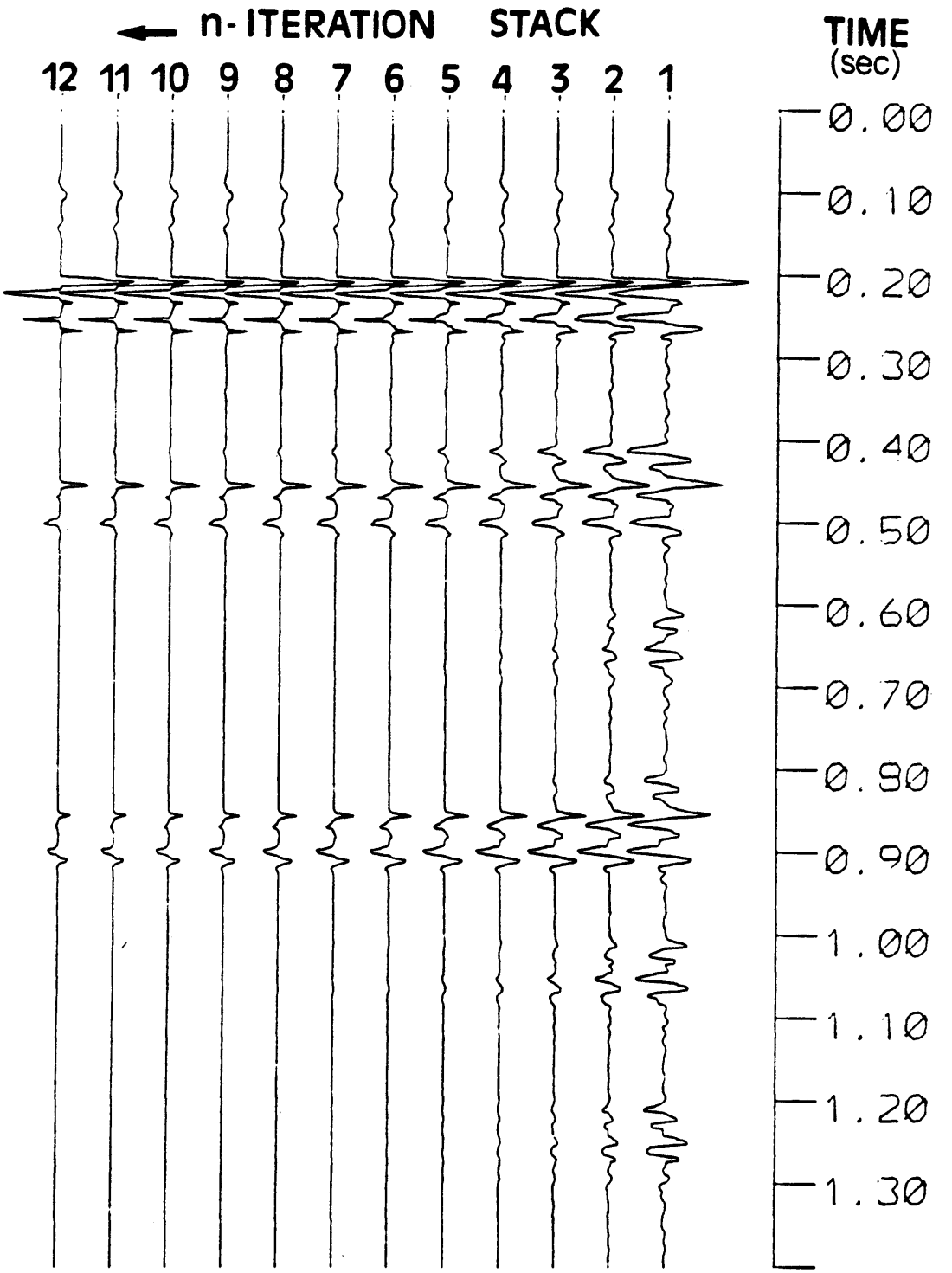


Figure 4-22, The final iterative stacks made by single-trace-processing plus a muting function.

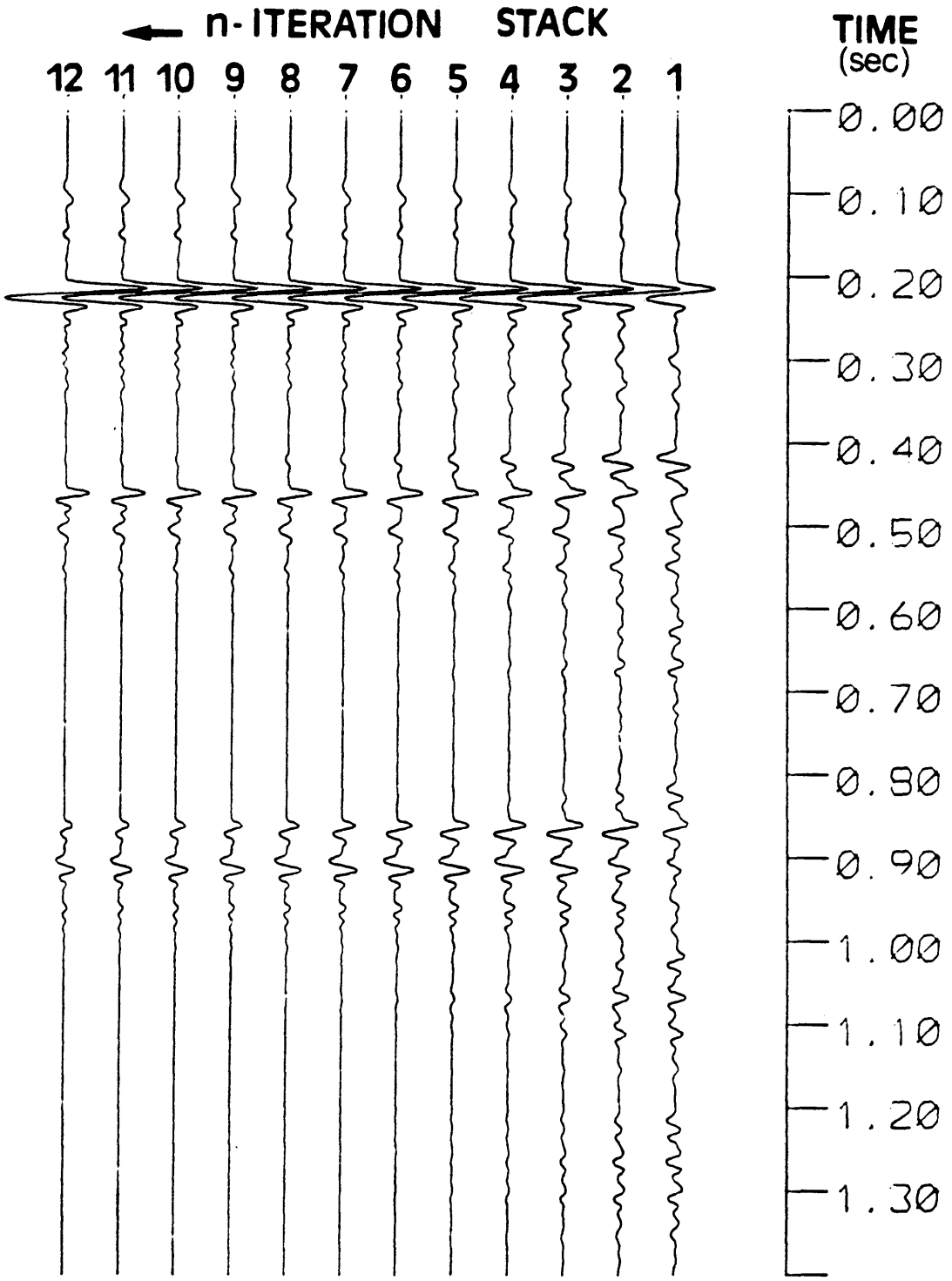


Figure 4-23, The result of Wiener process on fig. 4-22.

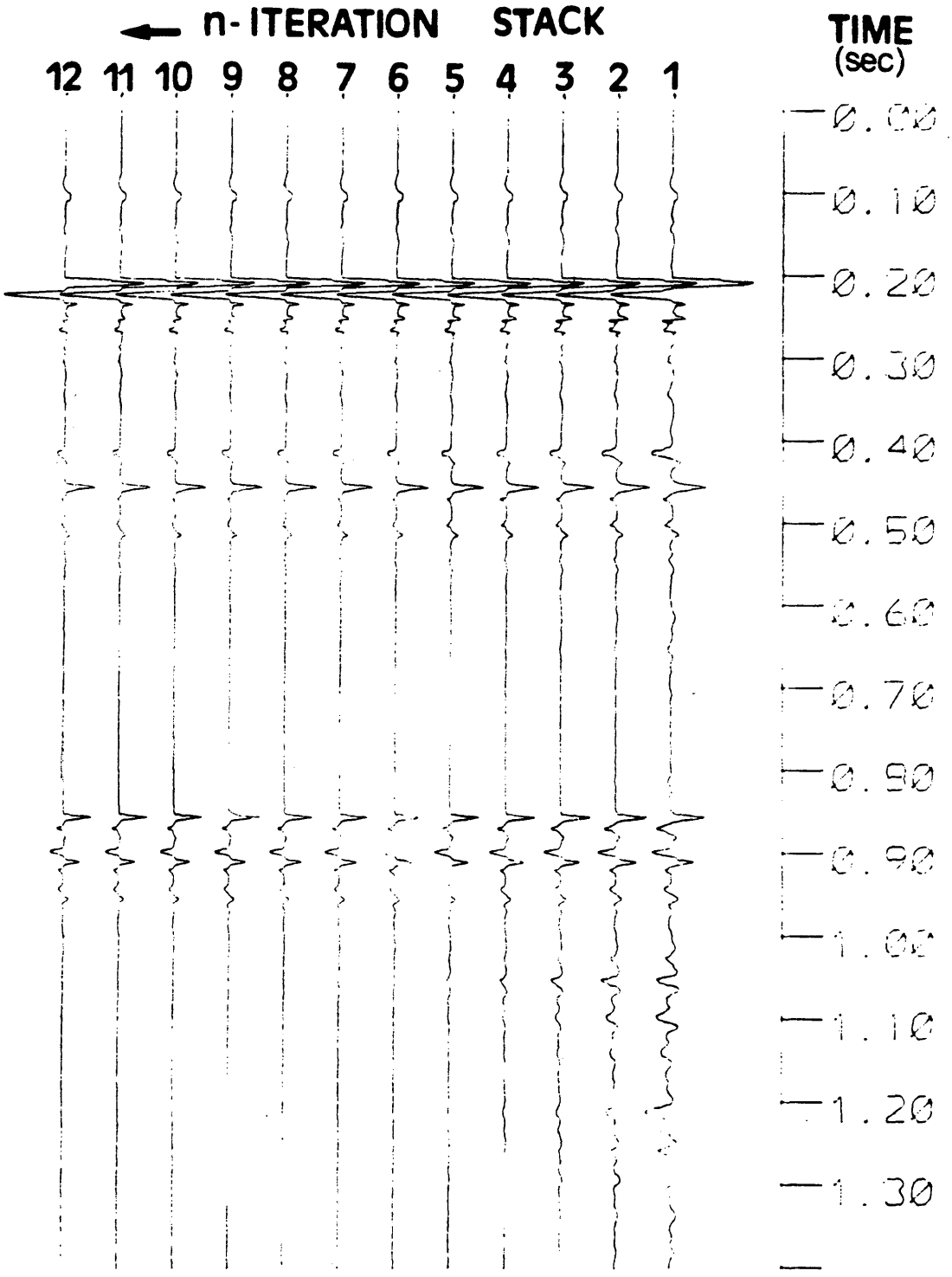


Figure 4-24, The result of predictive decon on fig. 4-22.

perform an iterative stacking to get a final stack as shown in figure 4-26, where the long-period multiples are deleted in this step. Figure 4-26 shows an almost perfect final section where both ghosts and multiples are almost totally suppressed.

It can be concluded that the iterative stacking (non-linear process) will prevent the subsequent Wiener process (linear process) from giving a normal performance, but, if we apply the Wiener process first it will not bother the subsequent iterative stacking. Thus, in routine procedure we suggest that the Wiener process should be executed prior to iterative stacking. This implies that in general linear processes should, when possible, procede the non-linear processes.

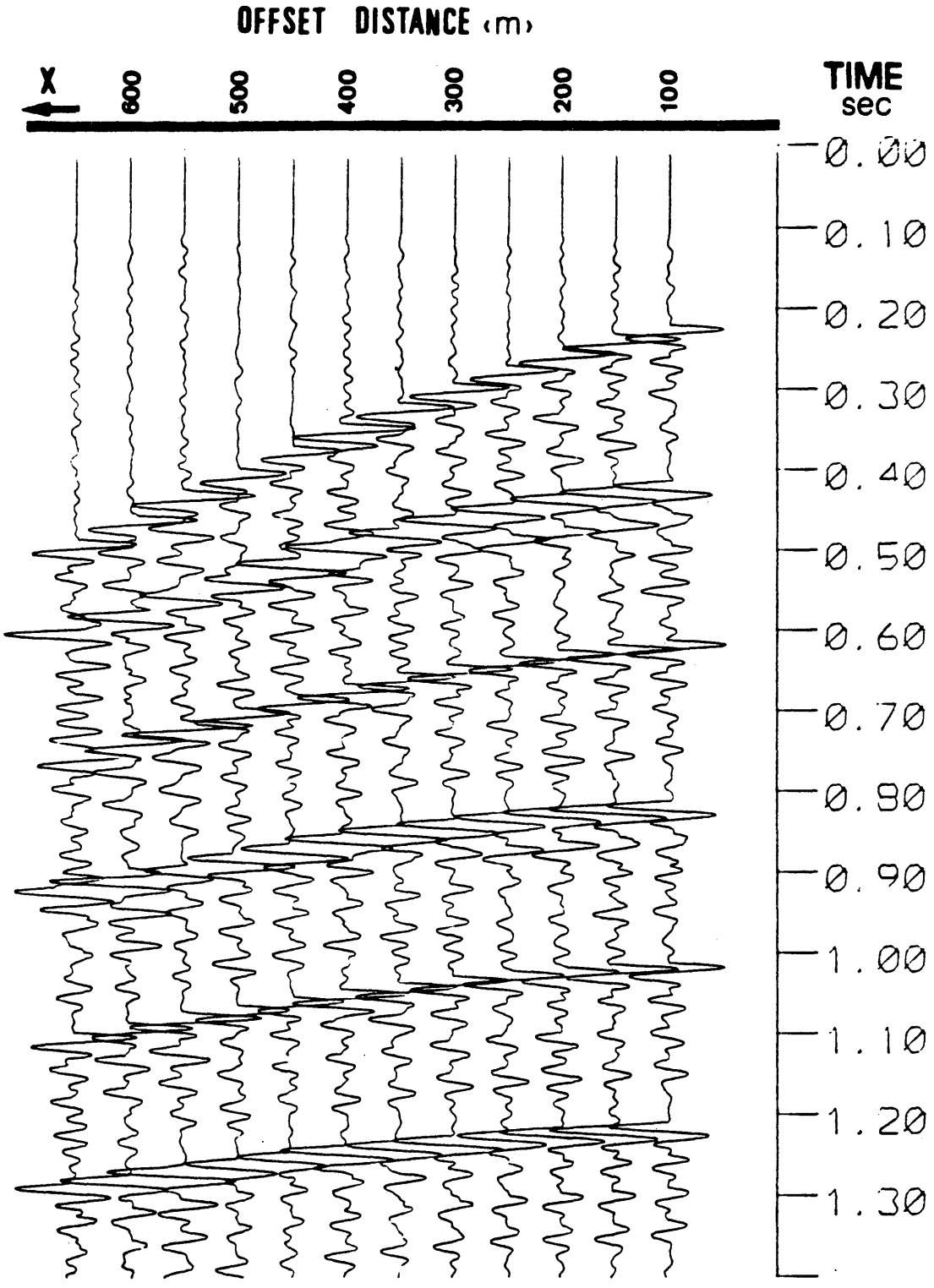


Figure 4-25, Prestack deghosting result.

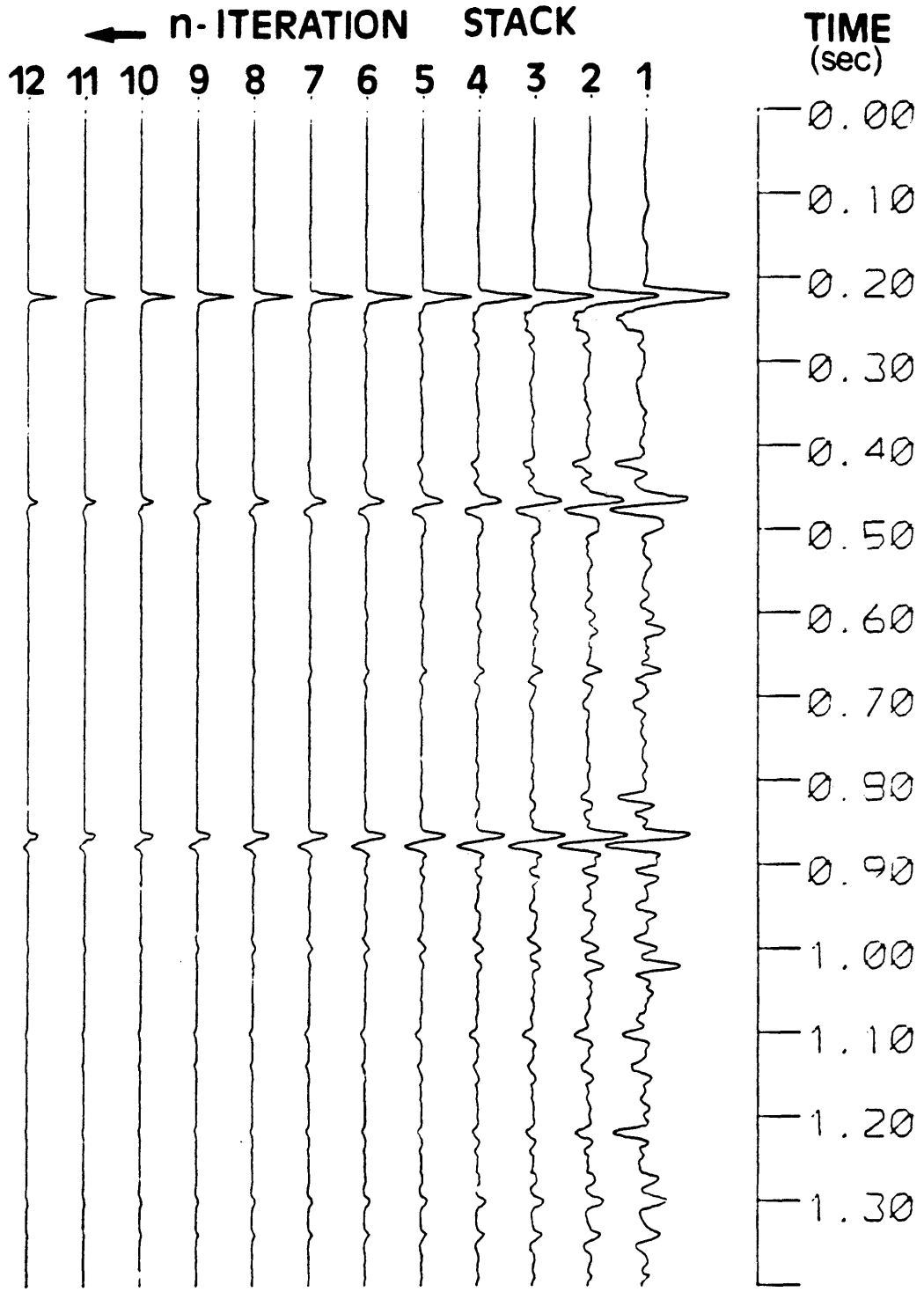


Figure 4-26, The final section after iterative process on figure 4-25.

5. CONCLUSIONS

From all of our modelling tests and comparisons, we have successfully fulfilled the objectives of understanding the characteristics of the iterative stacking algorithm, and of finding an accurate way to conduct the iterative stack in routine seismic processing. We summarized our conclusions as follows:

1) The iterative stack generally will be able to improve the primary to noise ratio because the algorithm takes into account the horizontal consistency of the amplitudes and not only the total straight sum. The iterative stack has proven to be a powerful tool in cleaning up the random noise, and is especially well adapted to suppressing strong long-period waterbottom multiples.

2) The iterative algorithm is a very flexible method. By varying the number of iterations and the muting parameters, it is possible to get a varying degree of emphasis on the horizontal consistency of the input. The algorithm may be considered as a generalized stacking method. In fact the straight stack is the 1-iteration version of the iteration output.

With the algorithm it is possible to output several versions. Routinely one may choose for comparison purposes, to display the first stack in addition to the iterative stacked sections. The iterative computation is so fast that the extra cost of doing this will be only the display costs.

3) The iterative algorithm may be applied anywhere after the linear seismic processing where a straight summation of traces is performed. Examples are velocity analysis, vertical stack, and the CDP stack. In velocity analysis, the iterative algorithm may resulted in improved velocity determination, whicn again will improve the final processed data.

4) For high resolution purposes, single-trace-processing is suggested for the output of the iterative process. The signal-to-noise ratio will be improved and the high frequency content will not be degraded as much.

5) It is suggested that muting should be included with the process of the iterative stack. The objective of muting is to alleviate the influence from NMO stretching, head waves, and the reflection amplitude change versus incidence angle.

These factors affect the ordinary stack as well as the iterative stack.

6) From our modelling tests, the iterative stack is generally more effective in suppressing background noise than the optimum weighted stack; moreover, the optimum weighted stack cannot achieve its full effectiveness in practice. These two algorithms are only partially compatible. If we put the iterative stack in routine processing, it is better to keep the manual editing and discard the diversity stack of the optimum weighted process.

7) The comparison of the iterative stack and f-k filtering has shown that both methods work well in deleting the horizontally travelling noise (low velocity events); but f-k filtering seems not to be able to clean up multiples as well as the iterative stack does. An example shows that these two methods are not compatible if conducted consectively. Aliasing and truncation problems which usually hamper the f-k transformation will make the combined output worse.

8) The iterative stack cannot suppress ghosts or water

reverberations. Deconvolution is still the best tool to suppress this noise. If the iterative stack is conducted prior the deconvolution, the Wiener operator could not be properly designed from stacked traces; muting and single-trace-processing may help reduce the error of the Wiener operator design for this approach. However, the much more accurate way is to do deconvolution prior to the iterative stack.

The examples show the disadvantages of the use of the iterative process. Iterative stacking changes the waveforms by its non-linear nature; the more iterations performed, the worse the distortion. An inference that can be drawn from this is that, when possible, linear processes should precede non-linear processes in the processing sequence.

9) Although some real data has been tested by Naess (1979), additional real data tests are still recommended. Other interesting topics are:

(i). The use of the iterative stack in the migration by summation method (Bortfeld 1974),

(ii). The susceptibility of the iterative stack to static errors.

In brief, iterative stacking is an exceptionally effective tool for the detection of coherent but weak signals; however it must be used with care. Most of our seismic data processing is based upon the approximate linearity of much of the physical process. If iterative stacking precedes such linearity-based processing it can invalidate the fundamental assumptions of the linearity-based processing.

6. REFERENCES

Bortfeld, R., 1974, Methods and Trends in Modern seismic exploration: Prakla-Seismos Publication.

Claerbout, J. F., 1976, Fundamentals of Geophysical Data Processing: with Applications to Petroleum Prospecting: New York, McGraw-Hill.

Dongara, J. J., Moler, C. B., Bunch, J.R., and Stewart G. W., 1979, Linpack Users' Guide: Society for Industrial and Applied Mathematics, Philadelphia.

George, C. F., Jr., Hall, S. D., and Slack, H. A., 1968, Coherence Measurements and Their Application in the Analysis of Seismic Data: Presented at the 38th Annual International SEG Meeting, Denver, Colorado.

GRAY Suite, 1983, The Generalized Ray Programs, Edited by Dr. F. Hadsell, Colorado School of Mines.

Green, P. E., Kelly, E. J., and Levin, M. J., 1966, A Comparison of Seismic Array Processing Methods: Geophys. J. Ray. Astr. Soc., 11, 64-84.

Lee, Y. W., 1960, Statistical Theory of Communication: New York, John Wiley and Sons, Inc.

Mayne, W. H., 1962, Common Reflection Point Horizontal Stacking Techneque: Geophysics 27, 92-114.

Naess, O. E., 1979, Superstack- an iterative stacking algorithm: Geophysical Prospecting 27, 16-28.

Naess, O. E., and Bruland, L., 1981, Velocity analysis using iterative stacking: Geophysical Prospecting 29, 1-20.

Naess, O. E., 1982, Single-trace processing using iterative CDP-stacking: Geophysical Prospecting 30, 641-652.

Neidell, N. S., and Taner, M. T., 1971, Semblance and other coherency measure for multichannel data: Geophysics 36, 482-497.

Ngian, F. H., 1983, Personal Communication.

Peacock, K. L., and Treitel, S., 1969, Predictive

Deconvolution, Theory and Practice: Geophysics 34, 155-169.

Rice, R. B., 1962, Inverse Convolution Filters: Geophysics 27, 4-18.

Robinson, E. A., 1966, Multichannel Z Transforms and Minimum Delay: Geophysics 31, 482-500.

Robinson, E. A., and Treitel, S., 1967, Principles ot Digital Wiener Filtering,: Geophysical Prospecting 15, 311-333.

Robinson, J. C., 1970, Statistically optimal stacking of seismic data: Geophysics 35, 436-446.

SEG 1974, Digital Processing of Geophysical Data-a Review, Continuing Education Program: Society of Exploration Geophysicists.

Sengbush, R. L., 1983, Seismic Exploration Methods: IHRDC Publishers, 137 Newbury Street, Boston, MA 02116.

Sheriff, R. E., 1982, Encyclopedia Dictionary of Exploration Geophysics. SEG, Tulsa, Oklahoma.

White, R. E., 1977, The performance of optimum stacking filters in suppressing uncorrelated noise: Geophysical Prospecting 25, 165-178.

7. APPENDICES

APPENDIX A

MAIN PROGRAMS

Main programs to produce a printout, an output file, and/or a plot. (time unit: second, length unit: meter)

- The Program 'STKTST'
- The Program 'YMOD'
- The Program 'XMOD'
- The Program 'FTMAIN'
- The Program 'DECON'
- The Program 'PDECON'
- The Program 'TRPLOT'
- The Program 'PLOFIL'

```
00010           PROGRAM STKTST
00020     C*****************************************************
00030     C
00040     C       STKTST TESTS THE PERFORMANCE OF THE
00050     C       ITERATIVE ALGORITHM.
00060     C
00070     C       ARRAY:
00080     C          X=10 RANDOM DATA.
00090     C
00100     C       YEN MAR-1984
00110     C*****************************************************
00120     C
00130           DIMENSION X(10),BUF(10)
00140           TYPE 150
00150       150 FORMAT(1X,'      10 RANDOM DATA TEST',/)
00160     C*******SET RANDOM DATA
00170           DO 1 I=1,10
00180           X(I)=RAN(N)
00190         1 CONTINUE
00200     C*******SET ITERATION TIMES
00210           NITER=20
00220     C*******MAKE ITERATIVE STACK AND PRINTOUT
00230           DO 60 ITER=1,NITER
00240           TPOS=0.0
00250           TNEG=0.0
00260           DO 70 J=1,N
00270           IF(X(J).GT. 0.0) TPOS=TPOS+X(J)
00280        70 IF(X(J).LT. 0.0) TNEG=TNEG+X(J)
00290           XN=N
00300           TPOS=TPOS/XN
00310           TNEG=TNEG/XN
00320           TT=TPOS+TNEG
00330           TYPE 100,X,TPOS
00340       100 FORMAT(1X,10F7.2,7X,'TPOS=',F5.2)
00350           TYPE 200,TNEG
00360       200 FORMAT(1X,77X,'TNEG=',F5.2)
00370           TYPE 300,ITER,TT
00380       300 FORMAT(1X,57X,I2,'-ITERATION OUTPUT--','SUM=',F5.2/)
00390           DO 75 J=1,N
00400           IF(X(J).GT.TPOS) X(J)=TPOS
00410        75 IF(X(J) .LT.TNEG) X(J)=TNEG
00420        60 CONTINUE
00430           STOP
00440           END
*
```

```
00010           PROGRAM YMOD
00020   C*********************************************************
00030   C
00040   C       YMOD GENERATES 12-FOLD CDP-GATHER FROM THE
00050   C       3-LAYER MODEL OR THE 4-LAYER MODEL.
00060   C
00070   C       ARRAYS:
00080   C          TC= 12-FOLD CDP-GATHER TRACES.
00090   C          X =OFFSET DISTANCE.
00100   C          XV=(X/VRMS)**2
00110   C          SW=SOURCE WAVELET.
00120   C          V =VRMS VELOCITY.
00130   C
00140   C       YEN MAR-1984
00150   C*********************************************************
00160           DIMENSION TC(700,12),X(12),XV(12),SW(40),V(700)
00170   C*******SET SINWAVE
00180           CALL SINWAV(41.,0.002,SW,25)
00190   C*******SET 3-LAYER MODEL
00200           CALL MODA(TC,X,XV)
00210   C*******YOU CAN ALSO CALL MODB FOR 4-LAYER MODEL.
00220   C*******ADD RANDOM NOISE
00230           CALL RAM(TC,0.06)
00240   C*******ADD GROUND NOISE
00250           CALL GROUND(TC,X)
00260   C*******CONVOLVE TC WITH THE WAVELET
00270           CALL CONVTC(TC,SW,700,12,25)
00280   C*******NMO CORRECTION
00290           CALL NMO(TC,V,X)
00300   C*******YOU CAN PLOT TC TO GET MODEL TRACES.
00310   C*******YOU CAN ALSO STORE TC INTO A FILE.
00320   C*******MAKE ITERATIVE STACK
00330           CALL STK(TC,12,2)
00340   C*******PLOT THE RESULT STACKS
00350           CALL PLO(TC,0.33,12,700,700)
00360           STOP
00370           END
*
```

```
00010           PROGRAM XMOD
00020   C*************************************************************
00030   C
00040   C       XMOD COMPUTES A MODEL OF 6-FOLD CDP-GATHER
00050   C       TRACES AND CALCULATES ITS OPTIMUM WEIGHTED STACK
00060   C       AND ITERATIVE STACK.
00070   C
00080   C       ARRAYS:
00090   C          TC=TRACE 1-6:  6-FOLD CDP-GATHER.
00100   C                   7:  IDEAL SIGNAL TRACE.
00110   C                   8:  STRAIGHT STACK.
00120   C                   9:  THE 4-ITERATION STACK.
00130   C                   10:  OPTIMUM WEIGHTED STACK.
00140   C                   11:  THE 4-ITERATION AFTER WEIGHT.
00150   C
00160   C       YEN MAR-1984
00170   C*************************************************************
00180           DIMENSION TC(280,11),BUF(280,7),SW(40)
00190           DIMENSION RATE(7),SCL(7)
00200   C*******SET SIGNAL SCALE
00210           DATA SCL/1.,1.,1.,1.,1.,1.,1./
00220   C*******SET RICKER WAVELET
00230           CALL RICKER(41.,0.002,SW,40)
00240   C*******SET MODELLING TRACE 1-6
00250           CALL MODX(TC,BUF,SW,RATE,SCL)
00260   C*******MAKE ITERATIVE STACK
00270           CALL STKGEN(BUF,280,6,6,2)
00280   C*******TRANSFER THE STRAIGHT STACK AND THE
00290   C       4-ITERATION STACK INTI TC TRACE 8 & 9
00300           DO 1 I=1,280
00310           TC(I,8)=BUF(I,1)
00320      1    TC(I,9)=BUF(I,4)
00330   C*******MAKE OPTIMUM WEIGHTING
00340           CALL WTD(TC,BUF,RATE,SCL)
00350   C*******MAKE ITERATIVE STACK ON WEIGHTED TRACES
00360           CALL STKGEN(BUF,280,6,6,2)
00370   C*******TRANSFER THE OPTIMUM WEIGHTED STACK AND
00380   C       THE 4-ITERATION WEIGHTED STACK INTO TC 10 & 11
00390           DO 2 I=1,280
00400           TC(I,10)=BUF(I,1)
00410      2    TC(I,11)=BUF(I,4)
00420   C*******PLOT THE 11 TRACES
00430           CALL PLOTX(TC,RATE,SCL)
00440           STOP
00450           END
*
```

```
00010           PROGRAM FTMAIN
00020     C*************************************************************
00030     C
00040     C       FTMAIN DOES F-K FILTERING AND PLOTS  THE RESULT.
00050     C       ITS INPUT IS READ FROM FOR02.DAT.
00060     C
00070     C       VARIABLES:
00080     C          DIP=DIPPING ANGLE IN DEGREES  FOR REJECT ZONE.
00090     C       ARRAYS:
00100     C          TC =12-FOLD CDP-GATHER.
00110     C
00120     C       YEN MAR-1984
00130     C*************************************************************
00140           DIMENSION TC(700,12)
00150     C*******SET REJECT DIP
00160           DIP=45.0
00170     C*******READ 12-FOLD TRACES FROM FILE FOR02.DAT
00180           DO 1 J=1,12
00190        1  READ(2,100) (TC(I,J),I=1,700)
00200     C*******EXEC F-K FILTERING
00210           CALL FT(TC,DIP,12)
00220       100 FORMAT(10E11.4)
00230     C*******PLOT THE RESULT
00240           CALL PLO(TC,0.33,12,512,700)
00250           STOP
00260           END
*
```

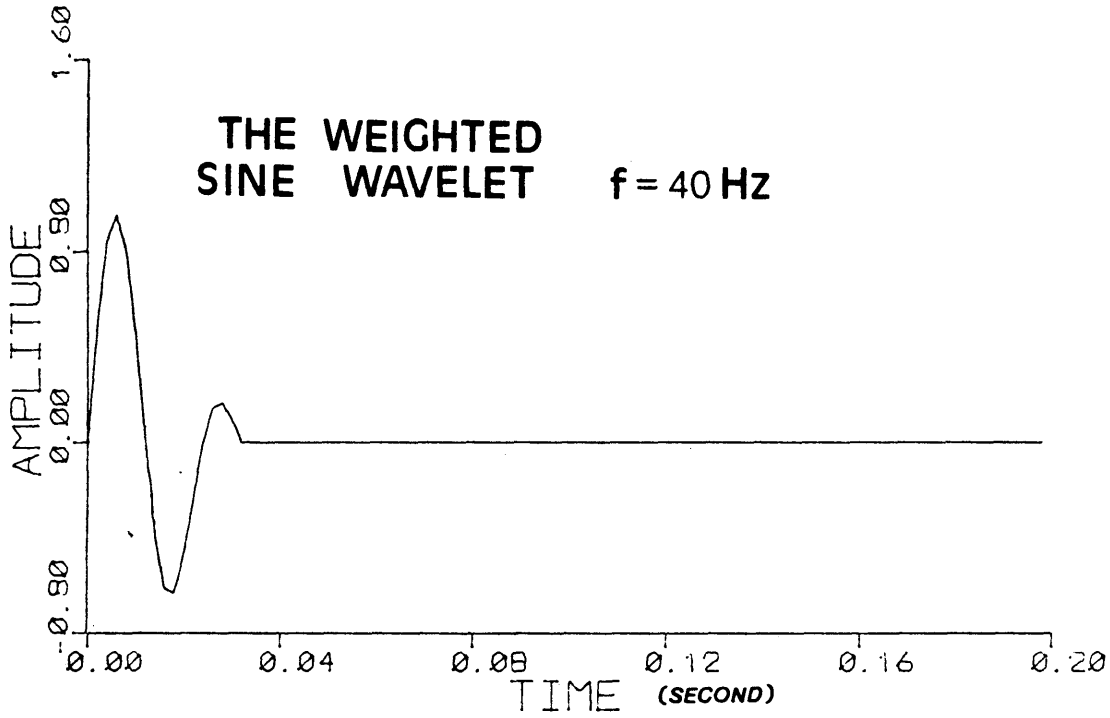
```
00010           PROGRAM DECON
00020     C*************************************************************
00030     C
00040     C       DECON APPLIES THE WIENER PROCESS TO INPUT ARRAY X.
00050     C       OUTPUT IS THE PLOT OF 12-FOLD DECON TRACES.
00060     C
00070     C       VARIABLES:
00080     C          LDA=LEADING DIMENSION OF MATRIX A.
00090     C          N=FILTER LENGTH.
00100     C          NZ=0, IF TRACES ARE NMO-CORRECTED.
00110     C             OTHEREWISE NOT YET.
00120     C       ARRAYS:
00130     C          X = INPUT ARRAY IN FILE FOR02.DAT WHICH
00140     C              CONTAINS 12-FOLD CDP-GATHER WITH 700
00150     C              DATA IN EACH TRACE; X WILL BE DECONVOLVED
00160     C              AFTER PROCESSED.
00170     C          A = MATRIX TO COMPUTE THE FILTER.
00180     C          Z = DESIRED OUTPUT TRACE.
00190     C          XX= AUTOCORRELATION OF X.
00200     C          ZX= CROSSCORRELATION OF Z AND X.
00210     C          IPVT=DUMMY FOR SUBROUTINE SGEFA & SGESL.
00220     C          TC= 12-FOLD CDP-GATHER.
00230     C
00240     C       YEN MAR-1984
00250     C*************************************************************
00260     C
00270           DIMENSION TC(700,12),A(120,120),SW(40)
00280           DIMENSION X(700),Z(700),XX(700),ZX(700),IPVT(120)
00290     C*******SET PARAMETERS
00300           LDA=120
00310           N=45
00320           NZ=0
00330     C*******SET SINWAVE
00340           CALL SINWAV(40.,0.002,SW,20)
00350           DO 1 L=1,12
00360     C*******READ IN ONE TRACE
00370           READ(2,100) (X(K)=1,700)
00380     C*******SET TRACE WINDOW
00390           CALL TRSET(X,L,NST,NW,NZ)
00400     C*******SET DESIRED OUTPUT
00410           Z(4)=1.0
00420     C*******CHECK THE WINDOW PARAMETERS
00430           TYPE 200,L,NST,NW
00440     C*******COMPUTE AUTOCORRELATION OF X
00450           CALL CROSS(X,X,XX,NW,NW)
00460     C*******COMPUTE CROSS CORRELATION OF ZX
00470           CALL CROSS(Z,X,ZX,NW,NW)
00480     C*******SET MATRIX
00490           CALL MXSET(XX,N,A,LDA)
00500     C*******SOLVE MATRIX
00510           CALL SGEFA(A,LDA,N,IPVT,INFO)
00520           IF(INFO.NE.0) GOTO 99
00530           CALL SGESL(A,LDA,N,IPVT,ZX,0)
00540     C*******SPIKE DECON
00550           CALL CONVXF(X,ZX,X,700,N,700)
00560           DO 3 I=1,700
00570        3  TC(I,L)=X(I)
00580        1  CONTINUE
00590           GOTO 999
```

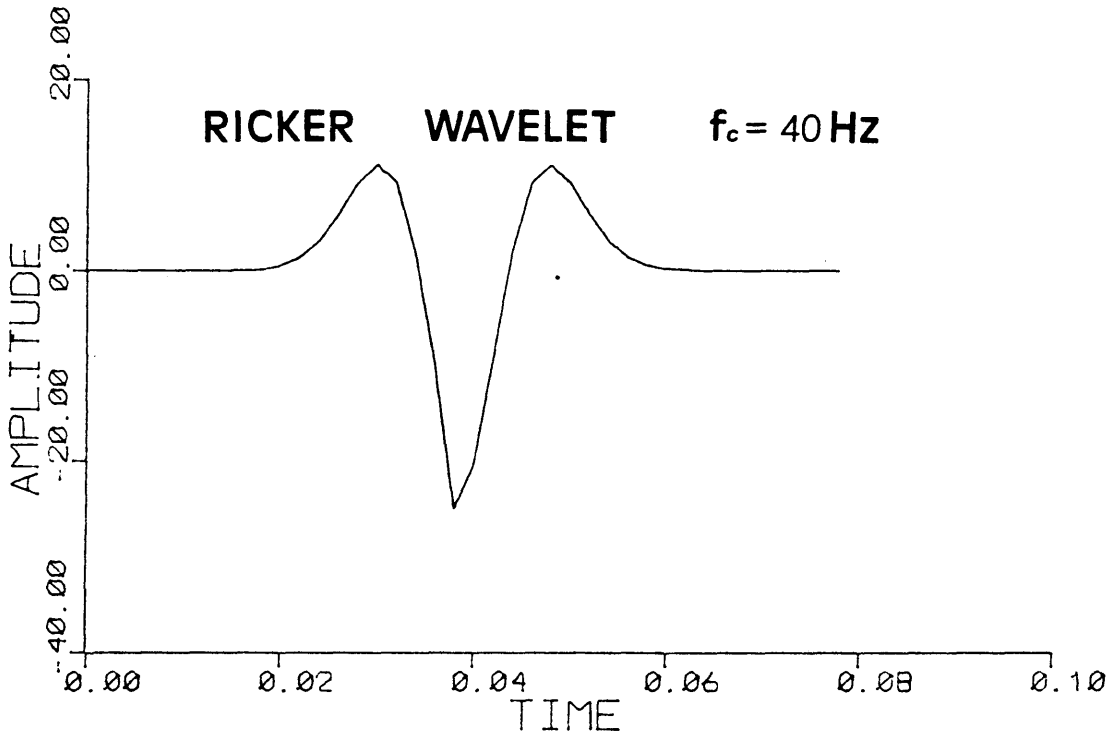
```
00600      99     TYPE 110
00610     110     FORMAT(1X,'INFO NOT 0')
00620     999     CONTINUE
00630    C*******CONVOLVE TC WITH KNOWN WAVELET
00640            CALL CONVTC(TC,SW,700,12,20)
00650            CALL PLO(TC,0.25,12,700,700)
00660     100    FORMAT(10E11.4)
00670      200   FORMAT(1X,'L=',I2,'NST=',I3,'NW=',I3)
00680            STOP
00690            END
*
```

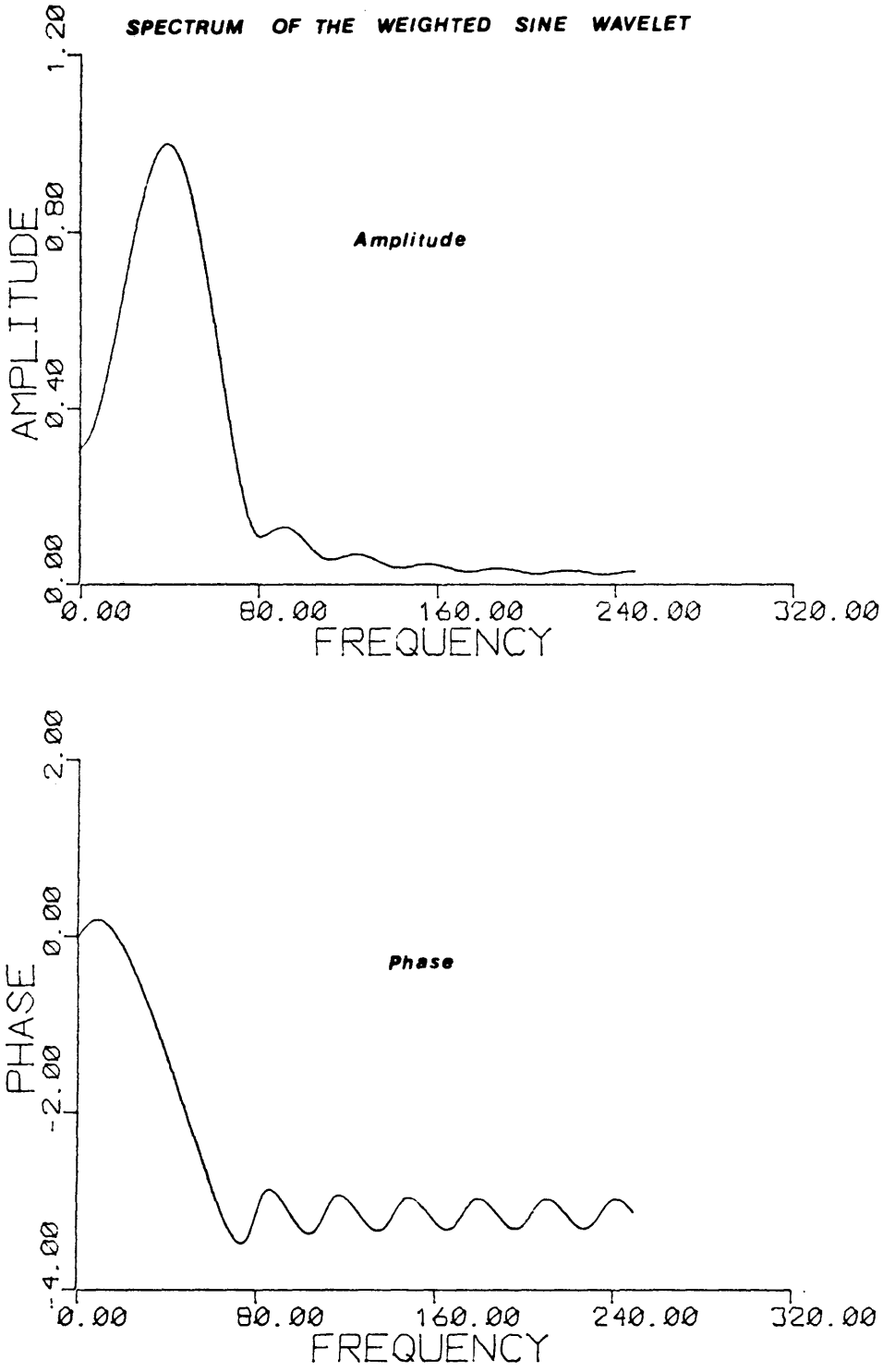
```
00010           PROGRAM PDECON
00020     C*************************************************************
00030     C
00040     C       PDECON DOES PREDICTIVE DECON ON INPUT ARRAY X.
00050     C       INPUT ARRAY READ FROM FOR02.DAT AND,
00060     C       OUTPUT IS THE PLOT OF 12-FOLD DECON TRACES.
00070     C
00080     C       VARIABLES:
00090     C          LDA=LEADING DIMENSION OF MATRIX A.
00100     C          N =LENGTH OF THE PREDICTIVE FILTER.
00110     C          NZ=0, IF TRACES ARE NMO-CORRECTED.
00120     C             OTHERWISE NOT YET.
00130     C          NR=PREDICTIVE DISTANCE.
00140     C          NF=LENGTH OF THE PREDICTIVE ERROR FILTER.
00150     C       ARRAYS:
00160     C          X = INPUT ARRAY IN FILE FOR02.DAT WHICH
00170     C              CONTAINS 12-FOLD CDP-GATHER WITH 700
00180     C              DATA IN EACH TRACE; X WILL BE DECONVOLVED
00190     C              AFTER PROCESSED.
00200     C          A = MATRIX TO COMPUTE THE FILTER.
00210     C          Z = DESIRED OUTPUT TRACE.
00220     C          XX= AUTOCORRELATION OF X.
00230     C          ZX= CROSSCORRELATION OF Z AND X.
00240     C          IPVT=DUMMY FOR SUBROUTINE SGEFA & SGESL.
00250     C          TC= 12-FOLD CDP-GATHER.
00260     C
00270     C       YEN MAR-1984
00280     C*************************************************************
00290     C
00300           DIMENSION TC(700,12),A(120,120)
00310           DIMENSION X(700),Z(700),XX(700),ZX(700),IPVT(120)
00320     C*******SET PARAMETERS
00330           LDA=120
00340           N=45
00350           NZ=0
00360           NR=20
00370           NF=N+NR
00380     C*******START PREDIC DECON
00390           DO 1 L=1,12
00400     C*******INPUT TRACE
00410           READ(2,100) (X(K)=1,700)
00420     C*******SET TRACE WINDOW
00430           CALL TRSET(X,L,NST,NW,NZ)
00440     C*******CHECK WINDOW PARAMETERS
00450           TYPE 200,L,NST,NW
00460     C*******COMPUTE AUTO X
00470           CALL CROSS(X,X,XX,NW,NW)
00480     C*******SET PREDICTIVE OUTPUT
00490           CALL PRDZX(XX,ZX,NW,NR)
00500     C*******SET MATRIX
00510           CALL MXSET(XX,N,A,LDA)
00520     C*******SOLVE MATRIX
00530           CALL SGEFA(A,LDA,N,IPVT,INFO)
00540           IF(INFO.NE.0) GOTO 99
00550           CALL SGESL(A,LDA,N,IPVT,ZX,0)
00560     C*******SET PRED ERROR FILTER
00570           CALL FSET(ZX,NR,N)
00580     C*******PREDIC DECON
00590           CALL CONVXF(X,ZX,X,700,NF,700)
```

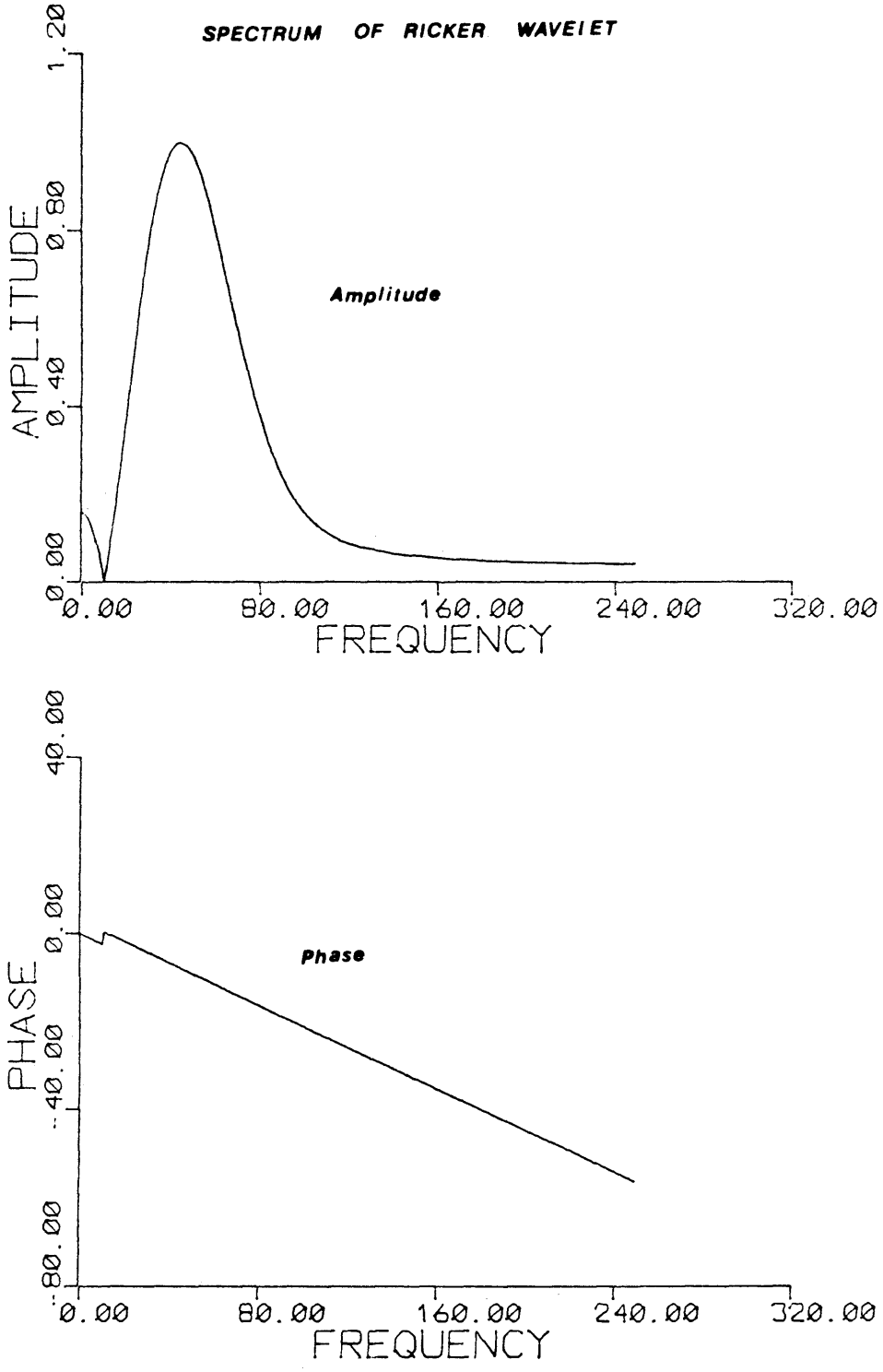
```
00600   C*******TRANSFER DECON TRACE TO TC
00610           DO 3 I=1,700
00620      3    TC(I,L)=X(I)
00630      1    CONTINUE
00640           GOTO 999
00650     99    TYPE 110
00660    110    FORMAT(1X,'INFO NOT 0')
00670    999    CONTINUE
00680   C*******PLOT TC
00690           CALL PLO(TC,0.25,12,700,700)
00700    100    FORMAT(10E11.4)
00710    200    FORMAT(1X,'L=',I2,'NST=',I3,'NW=',I3)
00720           STOP
00730           END
*
```

```
00010           PROGRAM TRPLOT
00020     C*************************************************************
00030     C
00040     C       TRPLOT PLOTS A TRACE IN TIME DOMAIN OR TRANSFORMS
00050     C       THE TRACE INTO FREQUENCY DOMAIN AND PLOTS
00060     C       ITS AMPLITUDE AND PHASE SPECTRUMS.
00070     C
00080     C       ARRAY:
00090     C          FILT=THE TRACE TO BE PLOTTED.
00100     C
00110     C       YEN MAR-1984
00120     C*************************************************************
00130     C
00140           DIMENSION FILT(700)
00150     C*******SET SINWAVE AND PLOT
00160           CALL SINWAV(40.0,0.002,FILT,16)
00170           CALL TRACE(100,FILT,0.002)
00180     C*******TRANSFORM AND PLOT ITS SPECTRUM
00190           CALL SPECPL(40,FILT,0.002)
00200     C*******SET RICKER WAVELET AND PLOT
00210           CALL RICKER(40.,0.002,FILT,40)
00220           CALL TRACE(100,FILT,0.002)
00230     C*******TRANSFORM AND PLOT ITS SPECTRUM
00240           CALL SPECPL(40,FILT,0.002)
00250           STOP
00260           END
*
```


THE WEIGHTED
SINE WAVELET
f = 40 Hz
AMPLITUDE
1.60
0.90
0.00
-0.90
0.00
0.04
0.08
0.12
0.16
0.20
TIME (SECOND)


RICKER WAVELET
fc = 40 Hz
AMPLITUDE
20.00
0.00
-20.00
-40.00
0.00
0.02
0.04
0.06
0.08
0.10
TIME


SPECTRUM OF THE WEIGHTED SINE WAVELET
Amplitude
AMPLITUDE
1.20
0.80
0.40
0.00
0.00
80.00
160.00
240.00
320.00
FREQUENCY
Phase
PHASE
2.00
0.00
-2.00
-4.00
0.00
80.00
160.00
240.00
320.00
FREQUENCY


SPECTRUM OF RICKER WAVELET
AMPLITUDE
1.20
0.80
0.40
0.00
Amplitude
0.00
80.00
160.00
240.00
320.00
FREQUENCY
PHASE
40.00
0.00
-40.00
-80.00
Phase
0.00
80.00
160.00
240.00
320.00
FREQUENCY

```
00010           PROGRAM PLOFIL
00020   C*******************************************************
00030   C
000 0   C       PLOFIL PLOTS A DATA FILE OF 12 TRACES
000: 1  C       IN INTERACTIVE MODE.
00060   C
00070   C       VARIABLES:
00080   C          NF=FILE NUMBER OF FOR00.DAT
00090   C          NPT=DATA POINT TO BE PLOTTED IN ONE TRACE.
00100   C          DX=SPACING BETWEEN TWO TRACES.
00110   C       ARRAYS:
00120   C          TC=ARRAY TO BE PLOTTED.
00130   C
00140   C       YEN MAR-1984
00150   C*******************************************************
00160           DIMENSION TC(700,12)
00170           TYPE 100
00180     100   FORMAT(1X,'INPUT FILE NUMBER')
00190           ACCEPT 150,NF
00200     150   FORMAT(I2)
00210           DO 1 J=1,12
00220      1    READ(NF,900) (TC(I,J), I=1,700)
00230           TYPE 200
00240     200   FORMAT(1X,'INPUT NPT')
00250           ACCEPT 250,NPT
00260     250   FORMAT(I3)
00270           TYPE 300
00280     300   FORMAT(1X,'INPUT DX')
00290           ACCEPT 350,DX
00300     350   FORMAT(F5.2)
00310           CALL PLO(TC,DX,12,NPT,700)
00320     900   FORMAT(10E11.4)
00330           STOP
00340           END
*
```

APPENDIX B

SUBROUTINES

Subroutines to support the execution of the main programs in modelling, processing, and plotting jobs.

THE LIST OF SUBROUTINES:

Modelling routines:

```
MODA(TC,X,XV)
MODB(TC,X,XV)
MODX(TC,BUF,SW,RATE,SCL)
RICKER(F,DT,FILT,NFP)
SINWAV(F,DT,SW,NFP)
RAM(TC,FACT)
GHOST(TC,ND,R)
GROUND(TC,X)
CONVTC(TC,SW,NTC,NL,NSW)
```

Processing routines:

```
CONVXF(A,FILT,C,NA,NFILT,NC)
CROSS(A,FILT,C,MA,MFILT)
CVS(TC,X,ITER,VSTR,VINC,VEND)
FFT(LX,CX,SIGNI)
FSET(F,NR,N)
FT(T,DIP,NL)
MXSET(XX,N,A,LDA)
NMO(TC,V,X)
PRDZX(XX,ZX,NX,NR)
SGEFA(A,LDA,N,IPVT,INFO)
SGESL(A,LDA,N,IPVT,B,JOB)
TRSET(X,L,NST,NW,NZ)
WTD(TC,BUF,RATE,SCL)
```

Iterative processing routines:

```
STK(TC,NITER,JOB)
STKGEN(TC,NPT,NN,NITER,JOB)
STKLEV(X,N,NITER,JOB)
```

Plotting routines:

```
PLO(T,DX,NL,NPT,LDM)
PLOLAB
PLOTX(T,RATE,SCL)
PLO2(T,DX,J)
SPECPL(NFP,FILT,DT)
TRACE(NFP,AMP,DT)
```

```
00010   C*********************************************************
00020   C
00030   C       SUBROUTINE CONVTC
00040   C
00050   C*********************************************************
00060   C
00070   C       CONVOLVES 2-D ARRAY TC WITH 1-D ARRAY SW.
00080   C
00090   C       VARIABLES:
00100   C          NTC=DATA POINTS OF EACH TRACE.
00110   C          NL =NUMBER OF TRACES TO BE CONVOLVED.
00120   C          NSW=DATA POINTS OF WAVELET.
00130   C
00140   C       ARRAYS:
00150   C          TC=12-FOLD CDP-GATHER TRACES, WHICH CONTAIN
00160   C             REFLECTIVITIES ON ENTRY, AND WILL
00170   C             BE SYNTHETIC TRACES ON RETURN.
00180   C          SW=SOURCE WAVELET.
00190   C          BUFF=BUFFER.
00200   C
00210   C       YEN MAR-1984
00220   C*********************************************************
00230
00240           SUBROUTINE CONVTC(TC,SW,NTC,NL,NSW)
00250           DIMENSION TC(NTC,1),SW(1),BUFF(700)
00260   C*******START CONVOLUTION
00270           DO 1 J=1,NL
00280           DO 2 I=2,NTC
00290   C*******SET WINDOW FOR MULTIPLICATION
00300           MT=NSW
00310           IA=I
00320           IB=1
00330           SUM=0.0
00340           IF(MT.GT.I) MT=I
00350   C*******MULTIPLY AND SUM INTO BUFFER
00360           DO 3 NN=1,MT
00370           SUM=SUM+TC(IA,J)*SW(IB)
00380           IA=IA-1
00390      3    IB=IB+1
00400      2    BUFF(I)=SUM
00410   C*******TRANSFER BUFFER TO TRACE TC.
00420           DO 4 I=1,NTC
00430      4    TC(I,J)=BUFF(I)
00440      1     CONTINUE
00450           TYPE 99
00460      99   FORMAT(1X,'CONV DONE')
00470           RETURN
00480           END
00490
*E
```

```
00010   C*************************************************************
00020   C
00030   C       SUBROUTINE CONVXF
00040   C
00050   C*************************************************************
00060   C
00070   C       CONVOLVES ARRAY A AND FILTER B.
00080   C
00090   C       VARIABLES:
00100   C          NA=LENGTH OF ARRAY A.
00110   C          NB=LENGTH OF FILTER B.
00120   C          NC=LENGTH OF RESULTED TRACE.
00130   C       ARRAYS:
00140   C          A=ARRAY A ON ENTRY.
00150   C          B=FILTER B ON ENTRY.
00160   C          C=TO BE RETURN.
00170   C          BUFF=BUFFER TO AVOID DAMAGE WHEN C IS A.
00180   C
00190   C       YEN MAR-1984
00200   C*************************************************************
00210           SUBROUTINE CONVXF(A,B,C,NA,NB,NC)
00220           DIMENSION A(1),B(1),C(1),BUFF(700)
00230   C*******SET LENGTH OF NC
00240           NC=NA+NB-1
00250           IF(NC.GT.700) NC=700
00260   C*******START CONVOLUTION
00270           DO 1 I=1,NC
00280   C*******SET CONVOLUTION WINDOW
00290           MT=NB
00300           IA=I
00310           IB=1
00320           SUM=0.0
00330           IF(MT .GT. I) MT=I
00340   C*******MULTIPLY AND SUM
00350           DO 2 NN=1,MT
00360           SUM=SUM+A(IA)*B(IB)
00370           IA=IA-1
00380      2    IB=IB+1
00390      1    BUFF(I)=SUM
00400   C*******TRANSFER FROM BUFFER TO C
00410           DO 3 I=1,NC
00420      3    C(I)=BUFF(I)
00430           RETURN
00440           END
*EU
```

```
00010   C*********************************************************
00020   C
00030   C       SUBROUTINE CROSS
00040   C
00050   C*********************************************************
00060   C
00070   C       COMPUTES CROSS CORRELATION OF A AND B.
00080   C
00090   C       VARIABLES:
00100   C          MA=NUMBER OF ELEMENTS IN A.
00110   C          MB=NUMBER OF ELEMENTS IN B.
00120   C       ARRAYS:
00130   C          A = ENTRY TRACE, A IS LONGER THAN B.
00140   C          B = ENTRY TRACE TO BE CROSS WITH A.
00150   C          C = RETURN TRACE WITH LENGTH OF MA.
00160   C          BUF=BUFFER IN CASE C IS A.
00170   C
00180   C       YEN MAR-1984
00190   C*********************************************************
00200           SUBROUTINE CROSS(A,B,C,MA,MB)
00210           DIMENSION A(1),B(1),C(1),BUF(700)
00220           IF(MA.GT.700) MA=700
00230   C*******START CONVOLUTION
00240           DO 1 I=1,MA
00250           IA=I
00260           IB=1
00270           SUM=0.0
00280   C*******MULTIPLY AND SUM
00290           DO 2 K=I,MB
00300           IF(IS.GT.700) GOTO 2
00310           SUM=SUM+A(IA)*B(IB)
00320           IA=IA+1
00330      2    IB=IB+1
00340           BUF(I)=SUM
00350      1    CONTINUE
00360   C*******TRANSFER BUFFER TO C
00370           DO 3 I=1,MA
00380      3    C(I)=BUF(I)
00390           RETURN
00400           END
*EU
```

```
00010   C*************************************************************
00020   C
00030   C       SUBROUTINE CVS
00040   C
00050   C*************************************************************
00060   C
00070   C       PROCESSES CONSTANT VELOCITY ANALYSES
00080   C       BY ITERATIVE STACKING AND PLOT.
00090   C
00100   C       VARIABLES:
00110   C          ITER=NUMBER OF ITERATIONS WHEN MAKE STACKING,
00120   C               THE OUTPUT WILL BE STRAIGHT STACK IF 1.
00130   C          VSTR=BEGINNING VELOCITY ON TEST.
00140   C          VINT=INCREMENT IN VELOCITY TEST.
00150   C          VEND=ENDING VELOCITY ON TEST.
00160   C       ARRAYS:
00170   C          TC=12-FOLD CDP-GATHER.
00180   C          X =OFFSET**2
00190   C          XNMO=ONE LEVEL NMO-CORRECTED DATA.
00200   C          TV=CVS TRACE TO BE PLOTTED.
00210   C
00220   C       YEN MAR-1984
00230   C*************************************************************
00240           SUBROUTINE CVS(TC,X,ITER,VSTR,VINC,VEND)
00250           DIMENSION TC(700,1),X(1),XNMO(12),TV(700)
00260   C*******COMPUTE HOW MANY CVS' TO BE TESTED
00270           NL=((VEND-VSTR)/VINC)+1
00280           DO 1 J=1,NL
00290   C*******SET CONSTANT VELOCITY
00300           VAG=(VSTR+(J-1)*VINC)**2
00310           DO 2 I=1,700
00320   C*******SET ZERO OFFSET TIME
00330           TZ=(0.002*I)**2
00340           DO 3 L=1,12
00350   C*******FIND OFFSET ARRIVING TIME
00360   C       AND MAKE NMO CORRECTION
00370           XV=X(L)/VAG
00380           NTX=SQRT(TZ+XV)*500.0+1
00390           IF(NTX.GT.700) NTX=700
00400      3    XNMO(L)=TC(NTX,L)
00410   C*******MAKE ITERATIVE STACK
00420           CALL STKLEV(XNMO,12,ITER,2)
00430   C*******TRANSFER ITER-STACK INTO TV
00440      2    TV(I)=XNMO(ITER)
00450   C*******PLOT TV
00460      1    CALL PLO2(TV,DX,J)
00470           CALL PLOT(0,0,+999)
00480           RETURN
00490           END
*EU
```

```
00010   C*********************************************************
00020   C
00030   C       SUBROUTINE FFT
00040   C
00050   C*********************************************************
00060   C
00070   C       COMPUTES A FAST FOURIER TRANSFORMATION.
00080   C
00090   C       VARIABLES:
00100   C          LX= NUMBER OF DATA POINTS IN ARRAY.
00110   C          SIGNI= +1. TO GO FROM TIME TO FREQUENCY,
00120   C                 -1 TO GO FROM FREQUENCY TO TIME.
00130   C       ARRAYS:
00140   C          CX= ARRAY TO BE TRANSFORMED.
00150   C
00160   C       CLEARBOUT 1976, AFTER BRENNER
00170   C*********************************************************
00180           SUBROUTINE FFT(LX,CX,SIGNI)
00190           COMPLEX CX(LX),CARG,CEXP,CW,CTEMP
00200           J=1
00210           SC=SQRT(1./LX)
00220           DO 30 I=1,LX
00230           IF(I.GT.J)GOTO 10
00240           CTEMP=CX(J)*SC
00250           CX(J)=CX(I)*SC
00260           CX(I)=CTEMP
00270      10   M=LX/2.
00280      20   IF(J.LE.M) GOTO 30
00290           J=J-M
00300           M=M/2
00310           IF(M.GE.1) GOTO 20
00320      30   J=J+M
00330           L=1
00340      40   ISTEP=2*L
00350           DO 50 M=1,L
00360           CARG=(0.,1.)*(3.14159265*SIGNI*(M-1))/L
00370           CW=CEXP(CARG)
00380           DO 50 I=M,LX,ISTEP
00390           CTEMP=CW*CX(I+L)
00400           CX(I+L)=CX(I)-CTEMP
00410      50   CX(I)=CX(I)+CTEMP
00420           L=ISTEP
00430           IF(L.LT.LX) GOTO 40
00440           RETURN
00450           END
*EU
```

```
00010   C*************************************************************
00020   C
00030   C       SUBROUTINE FSET
00040   C
00050   C*************************************************************
00060   C
00070   C       SETS PREDICTIVE ERROR FILTER.
00080   C
00090   C       VARIABLES:
00100   C          NR=PREDICTIVE LENGTH IN NUMBER OF DATA POINTS.
00110   C          N =LENGTH OF THE PREDICTIVE FILTER.
00120   C       ARRAYS:
00130   C          F =PREDICTIVE FILTER ON ENTRY.
00140   C             PREDICTIVE ERROR FILTER ON RETURN.
00150   C
00160   C       YEN MAR-1984
00170   C*************************************************************
00180           SUBROUTINE FSET(F,NR,N)
00190           DIMENSION F(1)
00200   C*******SHIFT VALUE DOWNAND TIMES -1.
00210           DO 1 I=1,N
00220           J=N+1-I
00230           JJ=J+NR
00240           F(JJ)=F(J)*(-1.0)
00250      1    F(J)=0.0
00260           IF(NR.EQ. 1) GOTO 3
00270           KK=NR-1
00280   C*******PUT ZERO IN THE GAP.
00290           DO 2 I=1,KK
00300           II=I+1
00310      2    F(II)=0.0
00320      3    F(1)=1.0
00330           RETURN
00340           END
*EU
```

```
00010   C*************************************************************
00020   C
00030   C        SUBROUTINE FT
00040   C
00050   C*************************************************************
00060   C
00070   C        MAKE F-K FILTERING ACCORDING TO
00080   C        THE DEGREE OF DIP ON ENTRY.
00090   C
00100   C        VARIABLES:
00110   C           DIP=THE SLOPE OF SLICING  IN DEGREES.
00120   C           NL=NUMBER OF FOLDS FOR CDP-GATHER.
00130   C        ARRAYS:
00140   C           T =12-FOLD CDP-GATHER.
00150   C           AA=TRANSFORMATION FORM OF THE CDP-GATHER.
00160   C           A =VERTICAL TRANSFORMABLE TRACE FOR FFT.
00170   C           B =HORIZONTAL TRANSFORMABLE TRACE FOR FFT.
00180   C
00190   C        YEN MAR-1984
00200   C*************************************************************
00210           SUBROUTINE FT(T,DIP,NL)
00220           DIMENSION T(700,1)
00230           COMPLEX AA(512,16),A(512),B(16)
00240   C*******CLEAR AA.
00250           DO 1 J=1,16
00260           DO 2 I=1,512
00270      2    AA(I,J)=CMPLX(0.0,0.0)
00280      1    CONTINUE
00290   C*******SET X-T FORM FOR TRANSFORMATION.
00300           DO 3 J=1,NL
00310           DO 4 I=1,512
00320           TT=T(I,J)
00330      4    AA(I,J)=CMPLX(TT,0.0)
00340      3    CONTINUE
00350   C*******F TRANSFORM.
00360           DO 5 J=1,16
00370           DO 6 I=1,512
00380      6    A(I)=AA(I,J)
00390           CALL FFT(512,A,1.)
00400           DO 7 I=1,512
00410      7    AA(I,J)=A(I)
00420      5    CONTINUE
00430   C*******K TRANSFORM.
00440           DO 8 I=1,512
00450           DO 9 J=1,16
00460     9     B(J)=AA(I,J)
00470           CALL FFT(16,B,1.)
00480           DO 10 J=1,16
00490      10   AA(I,J)=B(J)
00500      8    CONTINUE
```

```
00510   C*******ZERO-OUT IN F-K DOMAIN.
00520           DIP=DIP*3.141598/180.
00530           DIP=TAN(DIP)*256.0/9.0
00540           DO 11 J=2,16
00550           JJ=J
00560           IF(J.GT.9) JJ=18-J
00570           NN=JJ*DIP
00580           IF(NN.GT.256)NN=256
00590           DO 12 I=1,NN
00600           II=513-I
00610           AA(II,J)=CMPLX(0.0,0.0)
00620      12   AA(I,J)=CMPLX(0.0,0.0)
00630      11   CONTINUE
00640   C*******X TRANSFORM.
00650           DO 13 I=1,512
00660           DO 14 J=1,16
00670      14   B(J)=AA(I,J)
00680           CALL FFT(16,B,-1.0)
00690           DO 15 J=1,16
00700      15   AA(I,J)=B(J)
00710      13   CONTINUE
00720   C*******T TRANSFORM.
00730           DO 18 J=1,16
00740           DO 19 I=1,512
00750      19   A(I)=AA(I,J)
00760           CALL FFT(512,A,-1.0)
00770           DO 20 I=1,512
00780      20   AA(I,J)=A(I)
00790      18   CONTINUE
00800   C*******BACK TO ORIGINAL CDP-GATHER.
00810           DO 21 J=1,NL
00820           DO 22 I=1,512
00830      22   T(I,J)=REAL(AA(I,J))
00840      21   CONTINUE
00850           TYPE 99
00860      99   FORMAT(1X,'FT DONE')
00870           RETURN
00880           END
*
```

```
00010   C*******************************************************
00020   C
00030   C       SUBROUTINE GHOST
00040   C
00050   C*******************************************************
00060   C
00070   C       SETS GHOST RESPONSES.
00080   C
00090   C       VARIABLES:
00100   C          ND=NUMBER OF DATA POINTS FOR GHOST DELAY.
00110   C          R =REFLECTION COEFF OF THE GHOST.
00120   C       ARRAY:
00130   C          TC=12-FOLD CDP-GATHER.
00140   C
00150   C       YEN MAR-1984
00160   C*******************************************************
00170           SUBROUTINE GHOST(TC,ND,R)
00180           DIMENSION TC(700,1)
00190           DO 1 J=1,12
00200           DO 2 N=1,700
00210           I=701-N
00220           IGHOST=I+ND
00230           IF(IGHOST.GT.700) GOTO 2
00240           TC(IGHOST,J)=TC(IGHOST,J)+TC(I,J)
00250       2   CONTINUE
00260       1   CONTINUE
00270           RETURN
00280           END
*EU
```

```
00010   C*************************************************************
00020   C
00030   C       SUBROUTINE GROUND
00040   C
00050   C*************************************************************
00060   C
00070   C       GENERATES GROUNG TRAVELLING NOISE.
00080   C
00090   C       VARIABLES:
00100   C          K=GROUND EVENTS FROM 1 TO 4 WITH VELOCITIES
00110   C            400,450,500,550 METER/SEC RESPECTIVELY.
00120   C       ARRAYS:
00130   C          TC=12-FOLD CDP-GATHER ON ENTRY AND WITH
00140   C             GROUNG NOISE ADDED ON RETURN.
00150   C          X =OFFSET**2
00160   C
00170   C       YEN MAR-1984
00180   *************************************************************
00190   C
00200           SUBROUTINE GROUND(TC,X)
00210           DIMENSION TC(700,1),X(1)
00220           DO 1 J=1,12
00230           XX=X(J)**0.5
00240           DO 2 K=1,4
00250   C*******LOCATE THE GROUND NOISE ARRIVING TIME
00260           NN=XX/(350.+K*50.)*500.0
00270           IF(NN.GT.700) GOTO 2
00280   C*******ADD GROUND NOISE WITH STRENGTH OF 0.35
00290           TC(NN,J)=TC(NN,J)+0.35
00300      2    CONTINUE
00310      1    CONTINUE
00320           RETURN
00330           END
*EU
```

```
00010   C*********************************************************
00020   C
00030   C       SUBROUTINE MODA
00040   C
00050   C*********************************************************
00060   C
00070   C       SETS THE ARRIVING TIME AND THE REFLECTION
00080   C       COEFF FOR 12-FOLD CDP GATHER TRACES FROM THE
00090   C       THREE-LAYER MODEL AS FOLLOWS:
00100   C       LAYER 1:  WATERBOTTOM, V=1500, VRMS=1500, Z=150.
00110   C       LAYER 2:  BOUNDARY A,  V=2000, VRMS=1795, Z=400.
00120   C       LAYER 3:  BOUNDARY B,  V=3000, VRMS=2437, Z=1000.
00130   C
00140   C       VARIABLES:
00150   C          TX=ARRIVING TIME OF OFFSET X FOR WATERBOT.
00160   C          TZ=ZERO OFFSET ARRIVING TIME FOR WATERBOT.
00170   C          TXA=ARRIVING TIME FOR BOUNDARY A.
00180   C          TXB=ARRIVING TIME FOR BOUNDARY B.
00190   C       ARRAYS:
00200   C          TC=12-FOLD CDP-GATHER.
00210   C          X =(OFFSET DISTANCE)**2 FOR 12 STATIONS.
00220   C          XV=(OFFSET/VRMS)**2.
00230   C
00240   C       YEN MAR-1984
00250   C*********************************************************
00260           SUBROUTINE MODA(TC,X,XV)
00270           DIMENSION TC(700,1),X(1),XV(1)
00280           DO 10 I=1,700
00290           DO 20 L=1,12
00300      20   TC(I,L)=0.0
00310      10   CONTINUE
00320   C*******SET OFFSET AND XV
00330           DO 30 I=1,12
00340           X(I)=(100.0+(I-1)*50.0)**2
00350      30   XV(I)=X(I)/2250000.0
00360   C*******SET ARRIVING TIME FOR WATE LAYER
00370           DO 40 IM=1,6
00380           TZ=(IM*0.2)**2
00390           DO 50 L=1,12
00400           TX=SQRT(TZ+XV(L))
00410           I=TX*500.0+1
00420           IF(I.GT.700) I=700
00430      50    TC(I,L)=-1.0*(-1.0)**IM
00440      40   CONTINUE
00450   C*******SET ARRIVING TIME FOR PRIMARIES
00460           DO 60 I=1,12
00470           XVA=X(I)/3222025.0
00480           XVB=X(I)/5938969.0
00490           TXA=SQRT(0.2025+XVA)
00500           TXB=SQRT(0.7225+XVB)
00510           IA=TXA*500.0+1
00520           IB=TXB*500.0+1
00530           TC(IA,I)=0.1+TC(IA,I)
00540      99   FORMAT(1X,'MODA DONE')
00550      60   TC(IB,I)=TC(IB,I)+0.2
00560           TYPE 99
00570           RETURN
00580           END
*EU
```

```
00010   C*********************************************************
00020   C
00030   C       SUBROUTINE MODB
00040   C
00050   C*********************************************************
00060   C
00070   C       SETS THE ARRIVING TIME AND THE REFLECTION
00080   C       COEFF FOR THE 4-LAYER MODEL AS FOLLOWS:
00090   C       LAYER 1: V=1500, VRMS=1500, Z=157. TZ=0.21
00100   C       LAYER 2: V=1700, VRMS=1618, Z=247, TZ=0.50
00110   C       LAYER 3: V=2090, VRMS=1843, Z=418, TZ=0.90
00120   C       LAYER 4: V=2555, VRMS=2142, Z=703, TZ=1.45
00130   C       REFLEC. COEFF. RESPECTIVELY:
00140   C       R12=0.6, R23=0.1, R34=0.3, R45=0.1
00150   C
00160   C       ARRAYS:
00170   C          TC=12-FOLD CDP-GATHER.
00180   C          X =OFFSET**2.
00190   C          XV=X/VRMS**2.
00200   C
00210   C       YEN MAR-1984
00220   C*********************************************************
00230           SUBROUTINE MODB(TC,X,XV)
00240           DIMENSION TC(700,1),X(1),XV(1)
00250   C*******SET OFFSET AND XV.
00260           DO 30 I=1,12
00270           X(I)=(200.0+(I-1)*100.0)**2
00280      30   XV(I)=X(I)/2250000.0
00290   C*******SET ARRIVING TIME OF WATER LAYER.
00300           DO 40 IM=1,6
00310           TZ=(IM*0.21)**2
00320           DO 50 L=1,12
00330           TX=SQRT(TZ+XV(L))
00340           NBEG=TX*500.0+1
00350           IF(NBEG.GT.700) GOTO 50
00360           TC(NBEG,L)=TC(NBEG,L)+0.6
00370      50   CONTINUE
00380      40   CONTINUE
00390   C*******SET ARRIVING TIME FOR PRIMARIES.
00400           DO 60 I=1,12
00410           XVA=X(I)/1618**2
00420           XVB=X(I)/1843.0**2
00430           XVC=X(I)/2142.0**2
00440           TXA=SQRT(0.25+XVA)*500.0
00450           TXB=SQRT(0.81+XVB)*500.0
00460           TVC=SQRT(2.1025+XVC)*500.0
00470           NA=TXA+1
00480           NB=TXB+1
00490           NC=TXC+1
00500           IF(NA.GT.700) GOTO 60
00510           IF(NB.GT.700) GOTO 60
00520           IF(NC.GT.700) GOTO 60
00530           TC(NA,I)=TC(NA,I)+0.064
00540           TC(NB,I)=TC(NB,I)+0.19
00550           TC(NC,I)=TC(NC,I)+0.058
00560      60   CONTINUE
00570           RETURN
00580           END
*EU
```

```
00010   ************************************************************
00020   C
00030   C       SUBROUTINE MODX
00040   C
00050   C***********************************************************
00060   C
00070   C       SETS A MODEL OF 6 CDP-GATHER TRACES.
00080   C
00090   C       VARIABLES:
00100   C          TNGS=TOTAL ENERGY OF SIGNAL.
00110   C          TNGN=TOTAL ENERGY OF NOISE.
00120   C
00130   C       ARRAYS:
00140   C          TC=6-FOLD CDP-GATHER AND ITS STACKS.
00150   C          BUF=BUFFER OF TC.
00160   C          SW=WAVELET.
00170   C          RATE=SIGNAL-TO-NOISE ENERGY RATIO.
00180   C          SCL=SIGNAL SCALE.
00190   C
00200   C       YEN MAR-1984
00210   C***********************************************************
00220           SUBROUTINE MODX(TC,BUF,SW,RATE,SCL)
00230           DIMENSION TC(280,7),BUF(280,6),SW(40)
00240           DIMENSION RATE(7),SCL(7)
00250   C*******SET RANDOM NOISE INTO TC & BUF.
00260           TNGS=4.0
00270           DO 1 J=1,6
00280           TNGN=0.0
00290           DO 2 I=1,280
00300           YYY=(RAN(N)-0.5)*0.3
00310           TNGN=YYY*YYY+TNGN
00320           BUF(I,J)=YYY
00330      2    TC(I,J)=YYY
00340      1    RATE(J)=TNGS/TNGN
00350   C*******SET SIGNAL INTO TC & BUF.
00360           DO 3 J=1,7
00370           DO 4 KK=1,4
00380           K=20+(KK-1)*70
00390           BUF(K,J)=(BUF(K,J)+1.0)*SCL(J)
00400      4    TC(K,J)=BUF(K,J)
00410      3    CONTINUE
00420   C*******PRODUCE SYNTHETIC TRACES.
00430           CALL CONVTC(TC,SW,280,7,40)
00440           DO 5 J=1,6
00450           DO 6 I=1,280
00460      6    BUF(I,J)=TC(I,J)
00470      5    CONTINUE
00480           RETURN
00490           END
*EU
```

```
00010   C*********************************************************
00020   C
00030   C       SUBROUTINE MXSET
00040   C
00050   C*********************************************************
00060   C
00070   C       SETS A MATRIX OF AUTOCORRE X FOR THE
00080   C       DESIGN OF THE WIENER OPERATOR.
00090   C
00100   C       VARIABLES:
00110   C          N =DESIRED DIMENSION OF THE MATRIX.
00120   C          LDA=THE MAX MATRIX DIMENSION IN MAIN PROGRAM.
00130   C
00140   C       ARRAYS:
00150   C          A  =RESULTED MATRIX ON RETURN.
00160   C          XX =AUTOCORR X ON ENTRY.
00170   C          BUFF=BUFFER TRACE.
00180   C
00190   C       YEN MAR-1984
00200   C*********************************************************
00210           SUBROUTINE MXSET(XX,N,A,LDA)
00220           DIMENSION A(LDA,1),XX(1),BUFF(120)
00230   C*******SET THE FIRST ROW
00240           DO 1 J=1,N
00250           BUFF(J)=XX(J)
00260      1    A(1,J)=XX(J)
00270   C*******SET THE REST ROWS
00280           DO 2 I=2,N
00290           DO 3 J=2,N
00300           K=N-J+2
00310      3    BUFF(K)=BUFF(K-1)
00320           BUFF(1)=XX(I)
00330           DO 4 J=1,N
00340      4    A(I,J)=BUFF(J)
00350      2    CONTINUE
00360           RETURN
00370           END
*EU
```

```
00010   C*****************************************************************
00020   C
00030   C       SUBROUTINE NMO
00040   C
00050   C*****************************************************************
00060   C
00070   C       MAKES NMO CORRECTION FOR THE ARRAY TC.
00080   C
00090   C       VARIABLES:
00100   C          TX=ARRIVING TIME OF OFFSET X.
00110   C          TZ=ARRIVING TIME OF ZERO OFFSET.
00120   C       ARRAYS:
00130   C          TC=12-FOLD CDP-GATHER ON ENTRY, AND WILL BE
00140   C             RETURN AFTER NMO CORRECTION.
00150   C          NA=VELOCITY DATA OF BOUNDARY A FROM KNOWN MODEL.
00160   C          NB=VELOCCITY DATA OF BOUNDARY B FROM KNOWN MODEL.
00170   C          X =(OFFSET DISTANCE FOR EACH STATION)**2
00180   C          V =RMS VELOCITY FOR EACH DATA POINT.
00190   C
00200   C       YEN MAR-1984
00210   C*****************************************************************
00220           SUBROUTINE NMO(TC,V,X)
00230           DIMENSION TC(700,1),V(1),X(1),NA(12),NB(12)
00240           DATA NA/116,117,118,119,121,122,124,126,127,129,131,132/
00250           DATA NB/241,241,241,241,241,242,242,243,243,243,244,244/
00260           DO 5 I=1,12
00270      5    X(I)=(100.0+(I-1)*50.0)**2
00280           DO 50 J=1,12
00290           N1=NA(J)
00300           N2=NB(J)
00310   C SET RMS VELOCITY DOWN TO WATERBOTTOM.
00320           DO 10 I=1,N1
00330     10    V(I)=2250000.0
00340           DT=N1*0.002
00350           AA=V(N1)*DT
00360           KK=N1+1
00370   C SET RMS VELOCITY DOWN TO BOUNDARY A.
00380           DO 20 I=KK,N2
00390           K=I-N1
00400     20    V(I)=(AA+K*8000.0)/(DT+K*0.002)
00410           DT=N2*0.002
00420           KK=N2+1
00430           AA=V(N2)*DT
00440   C SET RMS VELOCITY DOWN TO BOUNDARY B.
00450           DO 30 I=KK,699
00460           K=I-N2
00470     30    V(I)=(AA+K*18000.0)/(DT+K*0.002)
00480           NPP=10
00490   C EXEC NMO.
00500           DO 60 I=10,695
00510           TZ=(0.002*(I-1))**2
00520           TX=SQRT(TZ+X(J)/V(I))*500.0
00530           NTX=TX+1
00540           IF(NTX.GT.700) GOTO 60
00550           IF(NTX.LT.NPP) NTX=NPP
00560           TC(I,J)=TC(NTX,J)
00570           NPP=NTX
00580      60   CONTINUE
00590      50   CONTINUE
00600           RETURN
00610           END
*
```

```
00010   C*************************************************************
00020   C
00030   C       SUBROUTINE PLO
00040   C
00050   C*************************************************************
00060   C
00070   C       PLOTS 2-DIMENSIONAL ARRAY T.
00080   C
00090   C       VARIABLES:
00100   C          DX=SPACING BETWEEN TWO TRACES.
00110   C          NL=NUMBER OF LINES TO BE PLOTTED.
00120   C          NPT=NUMBER OF DATA POINTS IN ONE LINE.
00130   C          LDM=LEADING DIMENSION IN MAIN PROGRAM OF T.
00140   C          YGD=LOCATION IN Y-AXIS FOR THE LABEL.
00150   C          XSP=LOCATION IN X-AXIS FOR THE TRACE BEGGIN.
00160   C          FPN=LABEL CONTENT.
00170   C          YM =PLOT INCREAMENT IN Y-AXIS.
00180   C          XM =PLOT INCREAMENT IN X-AXIS.
00190   C       ARRAYS:
00200   C          T  = NL-FOLD CDP-GATHER TRACES.
00210   C
00220   C       YEN MAR-1984
00230   C*************************************************************
00240           SUBROUTINE PLO(T,DX,NL,NPT,LDM)
00250           DIMENSION T(LDM,1)
00260   C*******FIND MAX VALUE AND NORMALIZING.
00270           AMAX=-1000.0
00280           DO 40 J=1,NL
00290           DO 42 I=1,NPT
00300           TT=ABS(T(I,J))
00310      42   IF(TT.GT.AMAX) AMAX=TT
00320      40   CONTINUE
00330           DO 44 J=1,NL
00340           DO 46 I=1,NPT
00350      46   T(I,J)=T(I,J)*0.5/AMAX
00360      44   CONTINUE
00370           IERR=-2
00380           CALL PLOTS(IERR,0)
00390   C*******PLOT LABEL.
00400           CALL PLOT(7.0,8.0,-3)
00410           CALL PLOT(0.0,-7.0,+2)
00420           DO 50 I=1,15
00430           YGD=-0.5*I+0.5
00440           CALL PLOT(0.0,YGD,3)
00450           CALL PLOT(0.25,YGD,2)
00460           YGD=YGD-0.12
00470           FPN=0.1*I-0.1
00480      50   CALL NUMBER(0.3,YGD,0.16,FPN,0.0,+2)
00490   C*******PLOT NLINE TRACES.
00500           DO 52 J=1,NL
00510           XSP=-0.33-DX*J
00520           CALL PLOT(XSP,0.0,+3)
00530           YM=0.0
00540           XM=XSP
00550           DO 54 I=1,NPT
00560           YM=YM-0.01
00570           XM=XSP+T(I,J)
00580      54   CALL PLOT(XM,YM,+2)
00590      52   CONTINUE
```

```
00600           TYPE 99
00610      99   FORMAT(1X,'PLOT FINISHED')
00620           CALL PLOT(XM,YM,999)
00630           RETURN
00640           END
```

```
00010   C*************************************************************
00020   C
00030   C       SUBROUTINE PLOLAB
00040   C
00050   C*************************************************************
00060   C
00070   C       INITIALS THE PLOT AND PLOTS THE LABEL
00080   C       FOR 0-1.4 SEC TIME AXIX.
00090   C
00100   C       YEN MAR-1984
00110   C*************************************************************
00120           SUBROUTINE PLOLAB
00130           IERR=-2
00140           CALL PLOTS(IERR,0)
00150   C*******PLOT LABEL.
00160           CALL PLOT(16.0,9.,-3)
00170           CALL PLOT(0.0,-8.,+2)
00180           DO 50 I=1,17
00190           YGD=-0.5*I+0.5
00200           CALL PLOT(0.0,YGD,3)
00210           CALL PLOT(0.25,YGD,2)
00220           YGD=YGD-0.12
00230           FPN=0.1*I-0.1
00240      50   CALL NUMBER(0.3,YGD,0.16,FPN,0.0,+2)
00250           RETURN
00260           END
*EU
```

```
00010   C*************************************************************
00020   C
00030   C       SUBROUTINE PLOTX
00040   C
00050   C*************************************************************
00060   C
00070   C       PLOTS THE 11 MODX TRACES WITH RATES AND SCL.
00080   C
00090   C       ARRAYS:
00100   C          TC:  11 TRACES TO BE PLOTTED.
00110   C          RATE : S/N RATIO OF THE 6 MODEL TRACES.
00120   C          SCL : SIGNAL SCALE OF THE 6 TRACES.
00130   C
00140   C       YEN MAR-1984
00150   C*************************************************************
00160           SUBROUTINE PLOTX(T,RATE,SCL)
00170           DIMENSION T(280,1),RATE(1),SCL(1)
00180   C*******NORMALIZATION OF THE DATA
00190           AMAX=-1000.0
00200           DO 40 J=1,11
00210           DO 42 I=1,280
00220           TT=ABS(T(I,J))
00230     42    IF(TT.GT.AMAX) AMAX=TT
00240     40    CONTINUE
00250           DO 44 J=1,11
00260           DO 46 I=1,280
00270     46    T(I,J)=T(I,J)*0.5/AMAX
00280     44    CONTINUE
00290   C*******START PLOTTING
00300           IERR=-2
00310           CALL PLOTS(IERR,0)
00320           CALL PLOT(7.0,8.0,-3)
00330           DO 52 J=1,11
00340           XSP=-0.33-0.33*J
00350           CALL PLOT(XSP,0.,+3)
00360           YM=0.0
00370           XM=XSP
00380           DO 54 I=1,280
00390           YM=YM-0.02
00400           XM=XSP+T(I,J)
00410     54    CALL PLOT(XM,YM,+2)
00420           IF(J .GT. 6) GOTO 52
00430           YM=YM-0.5
00440           CALL PLOT(XSP,YM,+3)
00450           FPN=RATE(J)
00460           CALL NUMBER(XSP,YM,0.1,FPN,90.0,2)
00470   C*******PLOT S/N RATE
00480           YM=YM-1.0
00490           FPN=SCL(J)
00500           CALL PLOT(XSP,YM,+3)
00510   C*******PLOT SIGNAL SCALE
00520           CALL NUMBER(XSP,YM,0.1,FPN,90.0,2)
00530     52    CONTINUE
00540           CALL PLOT(0,0,+999)
00550           RETURN
00560           END
*EU
```

```
00010   C*********************************************************
00020   C
00030   C       SUBROUTINE PLO2
00040   C
00050   C*********************************************************
00060   C
00070   C       PLOTS ONE TRACE DATA.
00080   C
00090   C       VARIABLES:
00100   C          DX=SPACING BETWEEN TWO TRACES.
00110   C          J =THE NUMBER OF THE TRACE IN MAIN PROGRAM.
00120   C       ARRAYS:
00130   C          T=TRACE TO BE PLOTED.
00140   C
00150   C       YEN MAR-1984
00160   C*********************************************************
00170           SUBROUTINE PLO2(T,DX,J)
00180           DIMENSION T(1)
00190   C*******NORMALIZATION
00200           AMAX=50.0
00210           DO 42 I=1,700
00220           TT=ABS(T(I))
00230           IF(TT.GT.AMAX) AMAX=TT
00240      42   CONTINUE
00250   C*******START PLOTTING
00260           DO 46 I=1,700
00270      46   T(I)=T(I)*0.7/AMAX
00280           XSP=-.33-DX*J
00290           CALL PLOT(XSP,0.,+3)
00300           YM=0.0
00310           XM=XSP
00320           DO 54 I=1,700
00330           YM=YM-0.01
00340           XM=XSP+T(I)
00350      54   CALL PLOT(XM,YM,+2)
00360           RETURN
00370           END
*EU
```

```
00010   C*******************************************************
00020   C
00030   C       SUBROUTINE PRDZX
00040   C
00050   C*******************************************************
00060   C
00070   C       SETS DESIRED OUTPUT FOR PREDICTIVE DECON.
00080   C
00090   C       VARIABLES:
00100   C          NX=NUMBER OF DATA POINTS IN THE TRACE WINDOW.
00110   C          NR=PREDICTIVE LENGTH.
00120   C       ARRAYS:
00130   C          XX=TRACE ON ENTRY.
00140   C          ZX=TRACE OF PREDICTIVE OUTPUT ON RETURN.
00150   C
00160   C       YEN MAR-1984
00170   C*******************************************************
00180           SUBROUTINE PRDZX(XX,ZX,NX,NR)
00190           DIMENSION XX(1),ZX(1)
00200           DO 1 I=1,NX
00210           II=NR+I
00220       1   ZX(I)=XX(II)
00230           RETURN
00240           END
*EU
```

```
00010   C*********************************************************
00020   C
00030   C       SUBROUTINE RAM
00040   C
00050   C*********************************************************
00060   C
00070   C       GENERATES RANDOM VALUES INTO ARRAY TC
00080   C       AND WEIGHTED RANDOM VALUES BY FACT.
00090   C
00100   C       VARIABLES:
00110   C          SUMS=SUM OF SIGNAL AMPLITUDES.
00120   C          SUMN=SUM OF RANDOM NOISE AMPLITUDES.
00130   C          YYY=RANDOM VALUE GENERATED FROM RAN(N)
00140   C              FUNCTION, VALUE RANGES FROM -.5 TO .5.
00150   C          SN =SIGNAL-TON-NOISE AMP RATIO
00160   C          FACT=WEIGHTED VALUE, IT WAS 0.06 BEFORE.
00170   C       ARRAYS:
00180   C          TC =12-FOLD CDP TRACES.
00190   C
00200   C       YEN MAR-1984
00210   C*********************************************************
00220           SUBROUTINE RAM(TC,FACT)
00230           DIMENSION TC(700,1)
00240   C*******SET SUM OF TOTAL SIGNAL ENERGY
00250           SUMS=0.01*12.
00260           SUMN=0.0
00270   C*******SET RANDOM VALUES AND GET ITS SUM
00280           DO 10 I=50,690
00290           DO 20 J=1,12
00300           YYY=RAN(N)-0.5
00310           SUMN=SUMN+(YYY*FACT)**2
00320      20   TC(I,J)=TC(I,J)+YYY*FACT
00330      10   CONTINUE
00340           SN=SUMS/(SUMN/640)
00350   C*******COMPUTE THE S/N AMP RATIO
00360           SN=SQRT(SN)
00370           TYPE 100,SN
00380           TYPE 99
00390      99   FORMAT(1X,'RAM DONE')
00400     100    FORMAT(1X,'AMPS/N=',F11.3)
00410           RETURN
00420           END
00430
*EU
```

```
00010   C*******************************************************
00020   C
00030   C       SUBROUTINE RICKER
00040   C
00050   C*******************************************************
00060   C
00070   C       GENERATES A RICKER WAVELET.
00080   C
00090   C       VARIABLES:
00100   C          DT=TIME STEP BETWEEN DATA POINTS.
00110   C          F =CENTER FREQUENCY.
00120   C          NFP=NUMBER OF POINTS FOR WAVELET.
00130   C          T=TIME IN SECONDS.
00140   C       ARRAYS:
00150   C          FILT=WAVELET.
00160   C
00170   C       YEN MAR-1984
00180   C*******************************************************
00190           SUBROUTINE RICKER(F,DT,FILT,NFP)
00200           DIMENSION FILT(1)
00210           PI=3.14159
00220           T=0.0
00230           FACT=PI*F
00240           I1=(NFP+1)/2
00250   C*******COMPUTE THE RIGHT-HALF WAVELET
00260           DO 1 I=I1,NFP
00270           R1=FACT*T
00280           R1SQ=R1*R1
00290           R2=1-2.*R1SQ
00300           R3=EXP(-R1SQ)
00310           FILT(I)=R2*R3*(-25.0)
00320      1    T=T+DT
00330   C*******COPY THE LEFT SYMMETRICAL PART
00340           I2=I1-1
00350           J=NFP
00360           DO 2 I=1,I2
00370           FILT(I)=FILT(J)
00380      2    J=J-1
00390           RETURN
00400           END
*
```

```
00010   C*************************************************************
00020   C
00030   C       SUBROUTINE SGEFA(A,LDA,N,IPVT,INFO)
00040   C
00050   C       SUBROUTINE SGESL(A,LDA,N,IPVT,B,JOB)
00060   C
00070   C*************************************************************
00080   C
00090   C       SUBROUTINES SOLVE LINEAR EQUATION AX=B.
00100   C       MAKE REFERENCE TO THE SOFTWARE LINPAC FOR CODES.
00110   C       WHEN EXEC USE LBY:LINPAC/LIBRARY.
00120   C
00130   C       ON ENTRY:
00140   C          A = A 2-D ARRAY WITH DIMENSION(LDA,N)
00150   C          LDA= THE LEADING DIMENSION OF THE MATRIX A.
00160   C          N = THE ORDER OF THE MATRIX A.
00170   C          JOB=IF 0, THE SYSTEM AX=B IS SOLVED.
00180   C              IF NON-ZERO, A(*)X=B IS SOLVED.
00190   C
00200   C       ON RETURN:
00210   C          A = CONTAINS IN ITS UPPER TRIANGLE AN
00220   C              UPPER TRIANGLE MATRIX.
00230   C          IPVT= INTEGER ARRAY OF DIMENSION N WITH PIVOT
00240   C                INFORMATION.
00250   C          INFO= AN INTEGER ARRAY, IF 0 THE PROCESS IS OK.
00260   C                THE VALUE OF INFO SHOULD BE CHECKED.
00270   C          B = CONTAINS THE SOLUTION, X.
00280   C
00290   C       AFTER LINPAC 1979
00300   C*************************************************************
*
```

```
00010   C*********************************************************
00020   C
00030   C       SUBROUTINE SINWAV
00040   C
00050   C*********************************************************
00060   C
00070   C       GENERATES  A WEIGHTED SINWAVE.
00080   C
00090   C       VARIABLES:
00100   C          DT=TIME STEP BETWEEN DATA POINTS IN SEC.
00110   C          F =FREQUENCY OF SINEWAVE.
00120   C          NFP=NUMBER OF POINTS FOR WAVELET ON RETURN.
00130   C          NW=NUMBER OF POINTS FOR WAVELET PERIOD.
00140   C          PERD=WAVELET PERIOD IN SEC.
00150   C       ARRAY:
00160   C          SW=SINEWAVE ON RETURN.
00170   C                .
00180   C       YEN MAR-1984
00190   C*********************************************************
00200           SUBROUTINE SINWAV(F,DT,SW,NFP)
00210           DIMENSION SW(1)
00220   C*******SET PERIOD AND THE DATA POINTS OF SINWAVE
00230           PERD=1.0/F
00240           NW=PERD/DT
00250           PI=3.1416
00260   C*******SET INCREMENT AND WEIGHT
00270           PSIN=(PI/180.)*(360./NW)
00280           PCOS=(PI/180.)*(90.0/NFP)
00290   C*******BUILD SINWAVE
00300           DO 10 I=1,NFP
00310           K=I-1
00320           GA=K*PSIN
00330           GB=K*PCOS
00340           SW(I)=SIN(GA)*COS(GB)
00350     10    CONTINUE
00360           RETURN
00370           END
*
```

```
00010   C*************************************************************
00020   C
00030   C       SUBROUTINE SPECPL
00040   C
00050   C*************************************************************
00060   C
00070   C       PLOTS THE AMPLITUDE AND THE PHASE SPECTRUMS
00080   C       FOR THE FILTER.
00090   C
00100   C       VARIABLES:
00110   C          NFP=NUMBER OF DATA POINTS OF THE FILTER.
00120   C          DT =TIME STEP BETWEEN TWO DATA.
```

```
00130   C       ARRAYS:
00140   C          A  =COMPLEX ARRAY FOR FFT PROCESS.
00150   C                IT IS THE FILTER ON ENTRY,
00160   C                AND IT WILL BE THE FREQ -DOMAIN ON RETURN.
00170   C          FREQ=FREQUENCY VALUE FOR PLOTTING PAIR.
00180   C          AMP =AMPLITUDE VALUE FOR PLOTTING PAIR.
00190   C          PHZ =PHASE VALUE FOR PLOTTING PAIR.
00200   C
00210   C       YEN MAR-1984
00220   C*************************************************************
00230           SUBROUTINE SPECPL(NFP,FILT,DT)
00240           COMPLEX A(1024)
00250           DIMENSION FILT(1),FREQ(514),AMP(514),PHZ(514)
00260   C*******FILL THE FILTER INTO A
00270           DO 1 I=1,NFP
00280           T=FILT(I)
00290           A(I)=CMPLX(T,0.0)
00300     1     CONTINUE
00310   C*******TRANSFORM
00320           CALL FFT(1024,A,1.0)
00330   C*******SET NYQUIST FREQUENCY
00340           ADD=1./(512*2*DT)
00350           FREQ(1)=0.0
00360           DO 2 I=2,512
00370     2     FREQ(I)=FREQ(I-1)+ADD
00380   C*******SET PHASE AND NORMALIZE AMP
00390           AMPMAX=-1E5
00400           DO 3 I=1,512
00410           AMP(I)=(REAL(A(I))**2+AIMAG(A(I))**2)**0.5
00420           IF(AMP(I).GT.AMPMAX) AMPMAX=AMP(I)
00430     3     PHZ(I)=ATAN2(AIMAG(A(I)),REAL(A(I)))
00440   C*******MAKE THE PHASE CONTINUOUS
00450           PJ=0.0
00460           DO 40 I=2,512
00470           IF(ABS(PHZ(I)+PJ-PHZ(I-1))-3.141593) 40,40,10
00480     10    IF(PHZ(I)+PJ-PHZ(I-1)) 20,40,30
00490     20    PJ=PJ+3.141593*2.
00500           GOTO 40
00510     30    PJ=PJ-3.141593*2.
00520     40    PHZ(I)=PHZ(I)+PJ
00530           DO 4 I=1,512
00540     4     AMP(I)=AMP(I)/AMPMAX
00550   C*******PLOT (FREQ VS. AMP) &(FREQ VS. PHZ)
00560           IERR=-2
00570           CALL PLOTS(IERR,0)
00580           CALL SCALE(AMP,3.,512)
00590           CALL SCALE(PHZ,3.,512)
00600           CALL SCALE(FREQ,4.,512)
00610           CALL AXIS(1.,5.,9HFREQUENCY,-9,4.,0.,FREQ(513),FREQ(514))
00620           CALL AXIS(1.,5.,9HAMPLITUDE,+9,3.,90.,AMP(513),AMP(514))
00630           CALL AXIS(1.,1.,9HFREQUENCY,-9,4.,0.,FREQ(513),FREQ(514))
00640           CALL AXIS(1.,1.,5HPHASE,+5,3.,90.,PHZ(513),PHZ(514))
00650           CALL PLOT(1.,1.,-3)
00660           CALL LINE(FREQ,PHZ,512,0,0,0)
00670           CALL PLOT(0.,4.,-3)
00680           CALL LINE(FREQ,AMP,512,0,0,0)
00690           CALL PLOT(0,0,+999)
00700           RETURN
00710           END
*
```

```
00010   C*********************************************************
00020   C
00030   C       SUBROUTINE STK
00040   C
00050   C*********************************************************
00060   C
00070   C       PROCESSES ITERATIVE STACK ON TC.
00080   C
00090   C       VARIABLES:
00100   C          NN=NUMBER OF TRACES FOR STK DECIDED BY MUTING.
00110   C          TPOS=SUM OF THE POSITIVE GROUP.
00120   C          TNEG=SUM OF THE NEGATIVE GROUP.
00130   C          JOB=0: ITERATIVE PROCESS BUT NOT STACK.
00140   C              1: SINGLE-TRACE-PROCESS AND REPLAE TC.
00150   C              2: ITERATIVE STACK AND REPLACE TC.
00160   C          NITER=ITERATIONS WHEN CHOOSE JOB=0.
00170   C                OTHERWISE IT IS 12.
00180   C       ARRAYS:
00190   C          TC=12-FOLD CDP-GATHER ON ENTRY.
00200   C             1-12TH ITERATION RESULT ON RETURN.
00210   C          X =BUFFER.
00220   C
00230   C       YEN MAR-1984
00240   C*********************************************************
00250           SUBROUTINE STK(TC,NITER,JOB)
00260           DIMENSION TC(700,1),X(12)
00270           DO 50 I=1,700
00280           GG=I
00290   C*******SET MUTING FACTOR
00300           NN=0.465*SQRT(GG)
00310           IF(NN.LT.2) NN=2
00320           IF(NN.GT.12) NN=12
00330   C*******PROCESS ITERATIVE STACK
00340           DO 60 ITER=1,NITER
00350           TPOS=0.0
00360           TNEG=0.0
00370           DO 70 J=1,NN
00380           IF(TC(I,J) .GT. 0.0) TPOS=TPOS+TC(I,J)
00390      70   IF(TC(I,J) .LT. 0.0) TNEG=TNEG+TC(I,J)
00400           TPOS=TPOS/NN
00410           TNEG=TNEG/NN
00420           DO 75 J=1,12
00430           IF(TC(I,J) .GT. TPOS) TC(I,J)=TPOS
00440           IF(TC(I,J) .LT. TNEG) TC(I,J)=TNEG
00450      75   CONTINUE
00460   C*******SELECT ITERATIVE OUTPUT
00470           IF(JOB.EQ.2) X(ITER)=TPOS+TNEG
00480           IF(JOB.EQ.1) X(ITER)=TC(I,1)
00490      60   CONTINUE
00500           IF(JOB.EQ.0) GOTO 50
00510   C*******TRANSFER THE WANTED OUTPUT TO TC
00520           DO 80 K=1,NITER
00530      80   TC(I,K)=X(K)
00540      50   CONTINUE
00550           RETURN
00560           END
*
```

```
00010   C*********************************************************
00020   C
00030   C       SUBROUTINE STKGEN
00040   C
00050   C*********************************************************
00060   C
00070   C       PROCESSES ITERATIVE STACK ON TC.
00080   C       THIS IS A GENERALIZED FORM WITHOUT MUTING.
00090   C
00100   C       VARIABLES:
00110   C          NN=NUMBER OF TRACES FOR STACKING.
00120   C          NPT=DATA POINTS OF ONE TRACE.
00130   C          TPOS=SUM OF THE POSITIVE GROUP.
00140   C          TNEG=SUM OF THE NEGATIVE GROUP.
00150   C          JOB=0: ITERATIVE PROCESS BUT NOT STACK.
00160   C              1: SINGLE-TRACE-PROCESS AND REPLAE TC.
00170   C              2: ITERATIVE STACK AND REPLACE TC.
00180   C          NITER=ITERATIONS WHEN CHOOSE JOB=0.
00190   C                OTHERWISE IT IS NN.
00200   C       ARRAYS:
00210   C          TC=NN-FOLD CDP-GATHER ON ENTRY.
00220   C             1-NN ITERATION RESULT ON RETURN.
00230   C          X =BUFFER.
00240   C
00250   C       YEN MAR-1984
00260   C*********************************************************
00270           SUBROUTINE STKGEN(TC,NPT,NN,NITER,JOB)
00280           DIMENSION TC(NPT,1),X(24)
00290   C*******PROCESS ITERATIVE STACK
00300           DO 50 I=1,NPT
00310           DO 60 ITER=1,NITER
00320           TPOS=0.0
00330           TNEG=0.0
00340           DO 70 J=1,NN
00350           IF(TC(I,J) .GT. 0.0) TPOS=TPOS+TC(I,J)
00360      70   IF(TC(I,J) .LT. 0.0) TNEG=TNEG+TC(I,J)
00370           TPOS=TPOS/NN
00380           TNEG=TNEG/NN
00390           DO 75 J=1,NN
00400           IF(TC(I,J) .GT. TPOS) TC(I,J)=TPOS
00410           IF(TC(I,J) .LT. TNEG) TC(I,J)=TNEG
00420      75   CONTINUE
00430   C*******SELECT OUTPUT
00440           IF(JOB.EQ.2) X(ITER)=TPOS+TNEG
00450           IF(JOB.EQ.1) X(ITER)=TC(I,1)
00460      60   CONTINUE
00470           IF(JOB.EQ.0) GOTO 50
00480   C*******TRANSFER WANTED OUTPUT TO TC
00490           DO 80 K=1,NITER
00500      80   TC(I,K)=X(K)
00510      50   CONTINUE
00520           RETURN
00530           END
*
```

```
00010   C*********************************************************
00020   C
00030   C       SUBROUTINE STKLEV
00040   C
00050   C*********************************************************
00060   C
00070   C       STKLEV MAKES ITERATIVE STACK ON ONE LEVEL DATA.
00080   C
00090   C       VARIABLES:
00100   C          N=DATA POINTS OF ONE LEVEL.
00110   C          NITER=ITERATIONS TO BE RUN WHEN JOB=0.
00120   C          JOB=0: ITERATIVE PROCESS WITHOUT REPLACING.
00130   C              =1: SINGLE-TRACE-PROCESS AND REPLACING.
00140   C              =2: ITERATIVE STACKING AND REPLACING.
00150   C       ARRAYS:
00160   C          X=ONE LEVEL DATAB FROM TRACE 1 TO N.
00170   C          BUF=BUFFER FOR STORING THE RESULT STACK.
00180   C
00190   C       YEN MAR-1984
00200   C*********************************************************
00210           SUBROUTINE STKLEV(X,N,NITER,JOB)
00220           DIMENSION X(1),BUF(12)
00230   C*******PROCESS ITERATIVE STACK
00240           DO 60 ITER=I,NITER
00250           TPOS=0.0
00260           TNEG=0.0
00270           DO 70 J=1,N
00280           IF(X(J).GT. 0.0) TPOS=TPOS+X(J)
00290      70   IF(X(J).LT. 0.0) TNEG=TNEG+X(J)
00300           TPOS=TPOS/N
00310           TNEG=TNEG/N
00320           DO 75 J=1,N
00330           IF(X(J).GT.TPOS) X(J)=TPOS
00340      75   IF(X(J) .LT.TNEG) X(J)=TNEG
00350   C*******SELECT OUTPUT
00360           IF(JOB.EQ.2) BUF(ITER)=TPOS+TNEG
00370           IF(JOB.EQ.1) BUF(ITER)=X(1)
00380      60   CONTINUE
00390           IF(JOB.EQ.0) GOTO 99
00400           DO 80 I=1,N
00410   C*******TRANSFER WANTED OUTPUT TO TC
00420      80   X(I)=BUF(I)
00430      99   CONTINUE
00440           RETURN
00450           END
*
```

```
00010   C*************************************************************
00020   C
00030   C       SUBROUTINE TRACE
00040   C
00050   C*************************************************************
00060   C
00070   C       PLOTS A TIME SERIES.
00080   C
00090   C       VARIABLES:
00100   C          NFP=DATA POINTS OF THE TIME TRACE.
00110   C          DT =TIME STEP BETWEEN TWO DATA.
00120   C       ARRAY:
00130   C          FREQ=TIME AXIX WITH INCREMENT OF DT.
00140   C          AMP=THE AMPLITUDE OF THE INPUT FILT.
00150   C
00160   C       YEN MAR-1984
00170   C*************************************************************
00180           SUBROUTINE TRACE(NFP,AMP,DT)
00190           DIMENSION AMP(702),FREQ(702)
00200   C*******SET TIME WINDOW
00210           DO 2 I=1,NFP
00220     2     FREQ(I)=(I-1)*DT
00230   C*******PLOT (FREQ VS. AMP)
00240           IERR=-2
00250           CALL PLOTS(IERR,0)
00260           CALL SCALE(AMP,3.,NFP)
00270           CALL SCALE(FREQ,5.,NFP)
00280           NA=NFP+1
00290           NB=NFP+2
00300           CALL AXIS(1.,1.,4HTIME,-4,5.,0.,FREQ(NA),FREQ(NB))
00310           CALL AXIS(1.,1.,9HAMPLITUDE,+9,3.,90.,AMP(NA),AMP(NB))
00320           CALL PLOT(1.,1.,-3)
00330           CALL LINE(FREQ,AMP,NFP,0,0,0)
00340           CALL PLOT(0,0,+999)
00350           RETURN
00360           END
*
```

```
00010   C*********************************************************
00020   C
00030   C       SUBRORUTINE TRSET
00040   C
00050   C*********************************************************
00060   C
00070   C       SES A WINDOW TRACE FOR THE WIENER PROCESS.
00080   C       THE WINDOW BEGINS AT THE FIRST REFLECTION,
00090   C       AND ENDS AT POINT 650.
00100   C
00110   C       VARIABLES:
00120   C          L =OFFSET LINE NUMBER ON ENTRY.
00130   C          NST=STARTING POINT OF THE FIRST REFLECTION.
00140   C          NW =WINDOW LENGTH.
00150   C          NZ = IF TRACE WAS NMO-CORRECTED NZ IS 0.
00160   C          XV=(OFFSET/VELOCITY)**2.
00170   C       ARRAYS:    .
00180   C          X =INPUT TRACE ON ENTRY,
00190   C             AND THE WINDOW TRACE ON RETURN,
00200   C             DATA AFTER NW ARE ZEROES.
00210   C
00220   C       YEN MAR-1984
00230   C*********************************************************
00240           SUBROUTINE TRSET(X,L,NST,NW,NZ)
00250           DIMENSION X(1)
00260   C*******FIND THE START POINT OF THE 1ST REFLECTION
00270           XV=(100.0+(L-1)*50.0)**2/2250000.0
00280           NST=1+500.0*SQRT(0.04+XV)
00290   C*******FIND THE WINDOW LENGTH
00300           NW=650-NST
00310           IF(NZ.EQ.0) NST=101
00320   C*******RESET THE INPUT TRACE
00330           DO 2 K=1,NW
00340           KK=K+NST
00350      2    X(K)=X(KK)
00360           KK=NW+1
00370   C*******ZERO THE TAIL
00380           DO 3 K=KK,700
00390      3    X(K)=0.0
00400           RETURN
00410           END
*
```

```
00010   C*********************************************************
00020   C
00030   C       SUBROUTINE WTD
00040   C
00050   C*********************************************************
00060   C
00070   C       MAKES OPTIMUM WEIGHTING ON TC.
00080   C
00090   C       VARIABLES:
00100   C          WT=OPTIMUM WEIGHTED RATIO.
00110   C       ARRAYS:
00120   C          TC=6-FOLD CDP-GATHER.
00130   C          BUF=BUFFER OF TC FOR WEIGHTED PROCESS.
00140   C          RATE=S/N ENERGY RATIO.
00150   C          SCL=SIGNAL SCALE.
00160   C
00170   C       YEN MAR-1984
00180   C*********************************************************
00190           SUBROUTINE WTD(TC,BUF,RATE,SCL)
00200           DIMENSION TC(280,1),BUF(280,1),RATE(1),SCL(1)
00210           DO 1 J=1,6
00220           GG=RATE(J)
00230           WT=SQRT(GG)/SCL(J)
00240           DO 2 I=1,280
00250      2    BUF(I,J)=TC(I,J)*WT
00260      1    CONTINUE
00270           RETURN
00280           END
*
```

APPENDIX C

Output of the Program 'GRAY'

THIS GENRAY FILE WAS CREATED 26-Jan-84
AT 06:08 HOURS

YOUR SOURCE-RECEIVER SECTION IS AS FOLLOWS:

MAX. NO. OF LEGS, NLEGS = 8
NO. OF LAYERS ABOVE LOWER HALFSPACE, NLAY = 3
SOURCE P=1, SV=2, SH=3: SOUR = 1
CONVERSIONS, REFL. ONLY=1, NONE=0, ALL=-1. CONV = 1
SOURCE DEPTH, HS = 0.100000E-01
FIRST RECEIVER, HR = 0.100000E-01 R = 100.0000
NR=12 RECEIVERS SEPARATED BY DR = 50.00000 DZ = 0.000

THE EARTH-MODEL YOU HAVE CONSTRUCTED LOOKS LIKE:

LAYER	VP	VS	DN	ZI
1	1.500	0.100	1.000	150.000
2	2.000	1.100	2.000	250.000
3	3.200	1.670	2.200	600.000
4	3.500	1.940	2.500	99.990

THE DISPLAY SECTION YOU HAVE SELECTED IS AS FOLLOWS:

SAMPLE INTERVAL = 2.000 (DELT) MILLISECONDS
INITIAL TIME = 100.000 (MNWND) MILLISECONDS
FINAL TIME = 1000.000 (MXWND) MILLISECONDS
SOURCE ENERGY = 1.000 (ENERGY) KILOJOULES.

THE MAXIMUM GAUSSIAN DISPLACEMENT ON A SPHERICAL SOURCE OF RADIUS
1.000 (SRAD) METERS IS .132E+04 MICRONS.
ALL PARTICLE DISPLACEMENTS CALCULATED IN MICRONS.

SUMMARY OF ARRIVAL TIMES AND AMPLITUDES FOR EACH RAY NAME

POSITIVE GROUND MOTION IS UP AND AWAY FROM THE SOURCE.
DIR: DIRECTION TAKEN BY RAY FROM THE SOURCE, 1 MEANS UP AND
-1 MEANS DOWN
ND: NUMBER OF DYNAMIC ANALOGS (SEE ENGELKEMEIR (T-2175))

THE SOURCE DEPTH IS 0.010 AND THE SOURCE IS IN LAYER # 1

RECEIVER # 1 IN LAYER # 1 , RECEIVER POSI : R= 100.0000 HR= 0.0100

SPECIES	ARRIVAL TIME	VERTICAL COMPO.	RADIAL COMP.	DIR	ND	RAYNAME
1	0.210846E+03	0.222968E+01	0.743276E+00	-1	1	P1P1
2	0.405584E+03	0.567[illegible]E+00	0.945277E-01	-1	1	P1P1P1P1
3	0.453421E+03	0.335[illegible]E+00	0.346102E-01	-1	1	P1P2P2P1
4	0.603679E+03	0.176[illegible]E+00	0.196254E-01	-1	1	P1P1P1P1P1P1P1
5	0.702424E+03	-0.223143E-01	-0.136457E-02	-1	1	P1P2P2P2P2P2P1
6	0.802399E+03	0.834817E-03	0.561836E-04	-1	1	P1P2P2P2P2P2P1
7	0.924599E+03	0.147938E-02	0.110710E-03	-1	1	P1P2S2S2P2P2P1
8	0.802399E+03	0.834817E-03	0.561836E-04	-1	1	P1P2S2S2P2P2P1
9	0.652613E+03	0.237091E+00	0.186941E-01	-1	2	P1P2P2P1P2P1
10	0.850976E+03	0.548313E-01	0.160329E-02	-1	1	P1P2P3P3P3P1
11	0.882762E+03	0.669110E-01	0.587626E-02	-1	1	P1P1P1P1P1P1P1P1P1P1
12	0.951435E+03	0.178201E-02	0.774647E-04	-1	1	P1P2P2P2P2P2P2P2P1
13	0.102266E+04	-0.307713E-04	-0.142971E-05	-1	1	P1P2P2P2P2S2P2P1
14	0.115377E+04	-0.494635E-04	-0.246652E-05	-1	1	P1P2P2P2P2S2S2P2P1
15	0.102260E+04	-0.307713E-04	-0.142971E-05	-1	1	P1P2P2P2P2S2P2P2P1
16	0.125496E+04	-0.972166E-04	-0.524392E-05	-1	1	P1P2P2P2S2S2S2P2P1
17	0.115377E+04	-0.595614E-04	-0.297455E-05	-1	1	P1P2P2P2S2S2P2P2P1
18	0.102260E+04	-0.307713E-04	-0.142971E-05	-1	1	P1P2P2P2S2P2P2P2P1
19	0.115377E+04	0.712169E-06	0.355545E-07	-1	1	P1P2P2P2S2P2S2P2P1
20	0.135618E+04	-0.161488E-03	-0.947521E-05	-1	1	P1P2S2P2S2S2S2P2P1
21	0.125496E+04	-0.972166E-04	-0.524392E-05	-1	1	P1P2S2P2S2S2P2P2P1
22	0.115377E+04	-0.492459E-04	-0.246652E-05	-1	1	P1P2S2P2S2P2P2P2P1
23	0.125496E+04	0.135786E-05	0.732514E-07	-1	1	P1P2S2S2P2S2S2P2P1
24	0.102260E+04	-0.307713E-04	-0.142971E-05	-1	1	P1P2S2P2P2S2P2P1
25	0.115377E+04	0.712169E-06	0.355545E-07	-1	1	P1P2S2P2P2S2S2P2P1
26	0.125496E+04	0.135786E-05	0.732514E-07	-1	1	P1P2S2P2S2S2S2P2P1
27	0.115377E+04	0.712169E-06	0.355545E-07	-1	1	P1P2S2P2S2P2S2P2P1
28	0.921769E+03	0.336451E-01	0.173923E-02	-1	1	P1P2P2P2P1P2P2P2P1
29	0.125856E+04	-0.198915E-02	-0.198276E-04	-1	1	P1P2P3P3P3P3P3P2P1
30	0.143088E+04	0.462729E-05	0.883565E-07	-1	1	P1P2P3P3S3P3P3P2P1
31	0.156926E+04	0.013416E-05	0.139399E-06	-1	1	P1P2P3S3S3P3P3P2P1
32	0.143088E+04	0.462729E-05	0.883565E-07	-1	1	P1P2P3S3P3P3P3P2P1
33	0.852111E+03	0.131923E+00	0.941544E-02	-1	1	P1P2P2P2P1P1P1P1P1
34	0.114033E+04	-0.122416E-01	-0.248816E-03	-1	2	P1P2P3P3P3P3P2P2P1
35	0.125319E+04	0.257825E-04	0.663334E-06	-1	1	P1P2P3P3P3S2P2P1
36	0.145243E+04	0.861868E-04	0.268898E-05	-1	1	P1P2P3S3S3S2P2P1
37	0.139132E+04	0.578555E-04	0.172481E-05	-1	1	P1P2P3S3S3P2P2P1
38	0.145093E+04	0.451[illegible]E-01	0.123[illegible]E-02	-1	2	P1P2P3P3P2P1P1P1

RECEIVER # 3 IN LAYER # 1 , RECEIVER POSI : R= 283.8688 HR= 0.0189

SPECIES	ARRIVAL TIME	VERTICAL COMPO.	RADIAL COMP.	DIR	ND	RAYNAME
1	0.240359E+03	0.143816E+01	0.958839E+00	-1	1	P1P1
2	0.421624E+03	0.471534E+00	0.157183E+00	-1	1	P1P1P1P1
3	0.463568E+03	0.289317E+00	0.591792E-01	-1	1	P1P2P2P1
4	0.613623E+03	0.158349E+00	0.351895E-01	-1	1	P1P1P1P1P1P1
5	0.798181E+03	-0.195475E-01	-0.238227E-02	-1	1	P1P2P2P2P2P2P1
6	0.829971E+03	0.295841E-02	0.396937E-03	-1	1	P1P2P2S2P2P2P1
7	0.912333E+03	0.539979E-02	0.761559E-03	-1	1	P1P2S2S2P2P2P1
8	0.829971E+03	0.295841E-02	0.396937E-03	-1	1	P1P2S2S2P2P2P1
9	0.661422E+03	0.214625E+00	0.337128E-01	-1	2	P1P2P2P1P1P1P1
10	0.855936E+03	0.526737E-01	0.321655E-02	-1	1	P1P2P3P3P3P2P1
11	0.811822E+03	0.564517E-01	0.948878E-02	-1	1	P1P1P1P1P1P1P1P1P1
12	0.955766E+03	0.160202E-02	0.139821E-03	-1	1	P1P2P2P2P2P2P2P2P1
13	0.125722E+04	-0.119895E-03	-0.102912E-04	-1	1	P1P2P2P2P2P2S2P2P1
14	0.115874E+04	-0.175314E-03	-0.174759E-04	-1	1	P1P2P2P2P2P2S2P2P1
15	0.125722E+04	-0.119895E-03	-0.102912E-04	-1	1	P1P2P2P2P2P2S2P2P1
16	0.126334E+04	-0.342118E-03	-0.368658E-04	-1	1	P1P2P2P2S2P2S2P2P1
17	0.115874E+04	-0.213534E-03	-0.212864E-04	-1	1	P1P2P2P2S2P2S2P2P1
18	0.125722E+04	-0.119895E-03	-0.102912E-04	-1	1	P1P2P2P2S2P2S2P2P1
19	0.115874E+04	0.103396E-04	0.103071E-05	-1	1	P1P2P2P2S2P2S2P2P1
20	0.136202E+04	-0.555412E-03	-0.951438E-04	-1	1	P1P2S2S2S2S2S2P2P1
21	0.126334E+04	-0.342118E-03	-0.368658E-04	-1	1	P1P2S2S2S2S2S2P2P1
22	0.115874E+04	-0.175314E-03	-0.174759E-04	-1	1	P1P2S2S2S2S2S2P2P1
23	0.126334E+04	0.194259E-04	0.299326E-05	-1	1	P1P2S2S2P2S2S2P2P1
24	0.125722E+04	-0.119895E-03	-0.122912E-04	-1	1	P1P2S2S2P2P2P2P2P1
25	0.115874E+04	0.103396E-04	0.103071E-05	-1	1	P1P2S2P2P2P2S2P2P1
26	0.126334E+04	0.194259E-04	0.299328E-05	-1	1	P1P2S2P2P2S2S2P2P1
27	0.115874E+04	0.103396E-04	0.103071E-05	-1	1	P1P2S2P2P2S2P2P2P1
28	0.966955E+03	0.314817E-01	0.324743E-02	-1	1	P1P2P2P1P1P2P2P2P1
29	0.125234E+04	-0.165779E-02	-0.369865E-03	-1	1	P1P2P3P3P3P3P2P2P1
30	0.141101E+04	0.181470E-04	0.682234E-06	-1	1	P1P2P3P3S3S2P3P2P2P1
31	0.157798E+04	0.739123E-04	0.121548E-05	-1	1	P1P2P3S3S3S2P3P2P2P1
32	0.141101E+04	0.181470E-04	0.682234E-06	-1	1	P1P2P3P3S3S2P3P2P2P1
33	0.855345E+03	0.172592E+00	0.156098E-01	-1	2	P1P2P3P1P1P1P1P1P1
34	0.113239E+04	-0.974528E-02	-0.482429E-03	-1	2	P1P2P3P3P2P2P2P2P1
35	0.122447E+04	0.160293E-03	0.513988E-05	-1	1	P1P2P3P3P2S2P2P1
36	0.146554E+04	0.332927E-03	0.207346E-04	-1	1	P1P2P3S3S2S2P2P2P1
37	0.136433E+04	0.224223E-03	0.133457E-04	-1	1	P1P2P3S3S2P2P2P2P1
38	0.125361E+04	0.441367E-01	0.239995E-02	-1	2	P1P2P3P3P2P1P1P1P1

RECEIVER # 5 IN LAYER # 1 , RECEIVER POSI : R= 303.3878 HR= 0.0188

SPECIES	ARRIVAL TIME	VERTICAL COMPO.	RADIAL COMP.	DIR	ND	RAYNAME
1	0.252853E+03	0.993627E+00	0.993693E+00	-1	1	P1P1
2	0.447560E+03	0.356429E+00	0.178211E+00	-1	1	P1P1P1P1P1
3	0.479448E+03	0.234783E+00	0.712231E-01	-1	1	P1P2P2P1
4	0.632443E+03	0.132963E+00	0.443219E-01	-1	1	P1P1P1P1P1P1P1
5	0.718137E+03	-0.157812E-01	-0.286796E-02	-1	1	P1P2P2P2P2P2P1
6	0.829964E+03	0.546996E-02	0.199547E-02	-1	1	P1P2P2S2P2P2P1
7	0.942225E+03	0.993378E-02	0.199547E-02	-1	1	P1P2S2S2P2P2P1
8	0.829964E+03	0.546996E-02	0.199547E-02	-1	1	P1P2S2S2P2P2P1
9	0.673219E+03	0.183902E+00	0.430572E-01	-1	2	P1P2P3P1P1P1P1
10	0.858444E+03	0.565184E-01	0.446111E-02	-1	1	P1P2P3P3P3P2P1
11	0.824688E+03	0.499748E-01	0.124669E-01	-1	1	P1P1P1P1P1P1P1P1P1
12	0.962938E+03	0.134548E-02	0.174486E-03	-1	1	P1P2P2P2P2P2P2P2P1
13	0.106489E+04	-0.211412E-03	-0.291864E-04	-1	1	P1P2P2P2P2P2S2P2P1
14	0.116697E+04	-0.322336E-03	-0.482118E-04	-1	1	P1P2P2P2P2P2S2P2P1
15	0.106489E+04	-0.211412E-03	-0.291864E-04	-1	1	P1P2P2P2P2P2S2P2P1
16	0.126922E+04	-0.622671E-03	-0.100337E-03	-1	1	P1P2P2P2S2P2S2P2P1
17	0.116697E+04	-0.400971E-03	-0.597936E-04	-1	1	P1P2P2P2S2P2S2P2P1
18	0.106489E+04	-0.211412E-03	-0.291864E-04	-1	1	P1P2P2P2S2P2S2P2P1
19	0.116697E+04	0.445843E-04	0.664849E-05	-1	1	P1P2P2P2S2P2S2P2P1
20	0.137166E+04	-0.967899E-03	-0.178113E-03	-1	1	P1P2S2S2S2S2S2P2P1
21	0.126922E+04	-0.622671E-03	-0.100337E-03	-1	1	P1P2S2S2S2S2S2P2P1
22	0.116697E+04	-0.322336E-03	-0.482118E-04	-1	1	P1P2S2S2S2S2S2P2P1
23	0.126922E+04	-0.815554E-04	-0.131544E-04	-1	1	P1P2S2S2P2S2S2P2P1
24	0.106489E+04	-0.211412E-03	-0.291864E-04	-1	1	P1P2S2S2P2P2P2P2P1
25	0.116697E+04	0.445843E-04	0.664849E-05	-1	1	P1P2S2S2P2P2S2P2P1
26	0.126922E+04	0.815554E-04	0.131544E-04	-1	1	P1P2S2S2P2P2S2P2P1
27	0.116697E+04	0.445843E-04	0.664849E-05	-1	1	P1P2S2S2P2P2S2P2P1
28	0.915363E+03	0.282844E-01	0.436282E-02	-1	1	P1P2P2P1P1P2P2P2P1
29	0.125510E+04	-0.143292E-02	-0.535477E-04	-1	1	P1P2P3P3P3P3P2P2P1
30	0.141498E+04	0.395179E-04	0.225666E-05	-1	1	P1P2P3P3P3P3P3P2P1
31	0.157491E+04	0.545411E-04	0.327659E-05	-1	1	P1P2P3P3S3S2P3P2P1
32	0.141498E+04	0.395179E-04	0.225666E-05	-1	1	P1P2P3P3P3P3P2P2P1
33	0.866987E+03	0.169514E+00	0.227483E-01	-1	3	P1P2P2P1P1P1P1P1P1
34	0.113739E+04	-0.927403E-02	-0.585991E-03	-1	2	P1P2P3P3P2P2P2P2P1
35	0.122674E+04	0.214346E-03	0.164552E-04	-1	1	P1P2P3P3P2S2P2P1
36	0.147711E+04	0.736847E-03	0.656261E-04	-1	1	P1P2P3S3S2S2P2P1
37	0.136924E+04	0.478792E-03	0.426194E-04	-1	1	P1P2P3S3S2P2P2P1
38	0.125813E+04	0.425263E-01	0.345547E-02	-1	2	P1P2P3P3P2P1P1P1

RECEIVER # 7 IN LAYER # 1 , RECEIVER POSI : R= 4[illegible] HR= [illegible]

SPECIES	ARRIVAL TIME	VERTICAL COMPO.	RADIAL COMP.	DIR	ND	RAYNAME
1	[illegible]	-[illegible]	-[illegible]	-1	1	P1P1
1	[illegible]	Arrival too close to reflect.		-1	1	P1P1P1
2	[illegible]	[illegible]	[illegible]	-1	1	P1P1P1P1
3	[illegible]	[illegible]	[illegible]	-1	1	P1P2P2P1
4	[illegible]	[illegible]	[illegible]	-1	1	P1P1P1P1P1
5	[illegible]	[illegible]	[illegible]	-1	1	P1P2P2P2P2P1
6	[illegible]	[illegible]	[illegible]	-1	1	P1P2P2S2P2P1
7	[illegible]	[illegible]	[illegible]	-1	1	P1P2S2S2F2P1
8	[illegible]	[illegible]	[illegible]	-1	1	P1P2S2F2P2P1
9	[illegible]	[illegible]	[illegible]	-1	2	P1P2P2P1P1P1
10	[illegible]	[illegible]	[illegible]	-1	1	P1P2P3F3P2P1
11	[illegible]	[illegible]	[illegible]	-1	1	P1P1P1P1P1P1P1P1
12	[illegible]	[illegible]	[illegible]	-1	1	P1P2P2P2P2P2P2P1
13	[illegible]	[illegible]	[illegible]	-1	1	P1P2P2P2P2S2P2P1
14	[illegible]	[illegible]	[illegible]	-1	1	P1P2P2P2S2S2P2P1
15	[illegible]	[illegible]	[illegible]	-1	1	P1P2P2P2S2S2P2P2P1
16	[illegible]	[illegible]	[illegible]	-1	1	P1P2P2S2S2S2S2P2P1
17	[illegible]	[illegible]	[illegible]	-1	1	P1P2P2S2S2S2P2P2P1
18	[illegible]	[illegible]	[illegible]	-1	1	P1P2P2S2S2F2P2P2P1
19	[illegible]	[illegible]	[illegible]	-1	1	P1P2P2S2F2S2S2P2P1
20	[illegible]	[illegible]	[illegible]	-1	1	P1P2S2S2S2S2S2P2P1
21	[illegible]	[illegible]	[illegible]	-1	1	P1P2S2S2S2S2P2P2P1
22	[illegible]	[illegible]	[illegible]	-1	1	P1P2S2S2S2F2P2P2P1
23	[illegible]	[illegible]	[illegible]	-1	1	P1P2S2S2F2S2S2P2P1
24	[illegible]	[illegible]	[illegible]	-1	1	P1P2S2S2F2S2P2P2P1
25	[illegible]	[illegible]	[illegible]	-1	1	P1P2S2F2S2S2S2P2P1
26	[illegible]	[illegible]	[illegible]	-1	1	P1P2S2F2S2S2P2P2P1
27	[illegible]	[illegible]	[illegible]	-1	1	P1P2S2F2S2F2S2P2P1
28	[illegible]	[illegible]	[illegible]	-1	1	P1P2S2F2S2F2P2P2P1
29	[illegible]	[illegible]	[illegible]	-1	1	P1P2P3F3P3F3P3P2P1
30	[illegible]	[illegible]	[illegible]	-1	1	P1P2P3F3S3P3P2P1
31	[illegible]	[illegible]	[illegible]	-1	1	P1P2P3S3S3P3P2P1
32	[illegible]	[illegible]	[illegible]	-1	1	P1P2P3S3F3P3P2P1
33	[illegible]	[illegible]	[illegible]	-1	3	P1P2P2P2F1F1P1P1P1
34	[illegible]	[illegible]	[illegible]	-1	2	P1P2P3F3F2P2P2P1
35	[illegible]	[illegible]	[illegible]	-1	1	P1P2P3F3F2S2P2P1
36	[illegible]	[illegible]	[illegible]	-1	1	P1P2P3S3F2S2P2P1
37	[illegible]	[illegible]	[illegible]	-1	1	P1P2P3S3F2P2P2P1
38	[illegible]	[illegible]	[illegible]	-1	2	P1P2P3F3P2P1P1P1

RECEIVER # 11 IN LAYER # 1 , RECEIVER POSI : R= 6[illegible] HR= [illegible]

SPECIES	ARRIVAL TIME	VERTICAL COMPO.	RADIAL COMP.	DIR	ND	RAYNAME
1	[illegible]	-[illegible]	-[illegible]	-1	1	P1P1
1	[illegible]	[illegible]	[illegible]	-1	1	P1H1P1
2	[illegible]	[illegible]	[illegible]	-1	1	P1P1P1P1
3	[illegible]	[illegible]	[illegible]	-1	1	P1P2P2P1
4	[illegible]	[illegible]	[illegible]	-1	1	P1P1P1P1P1P1
5	[illegible]	[illegible]	[illegible]	-1	1	P1P2P2P2P2P1
6	[illegible]	[illegible]	[illegible]	-1	1	P1P2P2S2P2P1
7	[illegible]	[illegible]	[illegible]	-1	1	P1P2S2S2P2P1
8	[illegible]	[illegible]	[illegible]	-1	1	P1P2S2F2P2P1
9	[illegible]	[illegible]	[illegible]	-1	2	P1P2P2P1P1P1
10	[illegible]	[illegible]	[illegible]	-1	1	P1P2P3F3P2P1
11	[illegible]	[illegible]	[illegible]	-1	1	P1P1P1P1P1P1P1P1
12	[illegible]	[illegible]	[illegible]	-1	1	P1P2P2P2P2P2P2P1
13	[illegible]	[illegible]	[illegible]	-1	1	P1P2P2P2P2S2P2P1
14	[illegible]	[illegible]	[illegible]	-1	1	P1P2P2P2S2S2P2P1
15	[illegible]	[illegible]	[illegible]	-1	1	P1P2P2P2S2S2P2P2P1
16	[illegible]	[illegible]	[illegible]	-1	1	P1P2P2S2S2S2S2P2P1
17	[illegible]	[illegible]	[illegible]	-1	1	P1P2P2S2S2S2P2P2P1
18	[illegible]	[illegible]	[illegible]	-1	1	P1P2P2S2S2F2P2P2P1
19	[illegible]	[illegible]	[illegible]	-1	1	P1P2P2S2F2S2S2P2P1
20	[illegible]	[illegible]	[illegible]	-1	1	P1P2S2S2S2S2S2P2P1
21	[illegible]	[illegible]	[illegible]	-1	1	P1P2S2S2S2S2P2P2P1
22	[illegible]	[illegible]	[illegible]	-1	1	P1P2S2S2S2F2P2P2P1
23	[illegible]	[illegible]	[illegible]	-1	1	P1P2S2S2F2S2S2P2P1
24	[illegible]	[illegible]	[illegible]	-1	1	P1P2S2S2F2S2P2P2P1
25	[illegible]	[illegible]	[illegible]	-1	1	P1P2S2F2S2S2S2P2P1
26	[illegible]	[illegible]	[illegible]	-1	1	P1P2S2F2S2S2P2P2P1
27	[illegible]	[illegible]	[illegible]	-1	1	P1P2S2F2S2F2S2P2P1
28	[illegible]	[illegible]	[illegible]	-1	1	P1P2S2F2S2F2P2P2P1
29	[illegible]	[illegible]	[illegible]	-1	1	P1P2P3F3P3F3P3P2P1
30	[illegible]	[illegible]	[illegible]	-1	1	P1P2P3F3S3P3P2P1
31	[illegible]	[illegible]	[illegible]	-1	1	P1P2P3S3S3P3P2P1
32	[illegible]	[illegible]	[illegible]	-1	1	P1P2P3S3F3P3P2P1
33	[illegible]	[illegible]	[illegible]	-1	3	P1P2P2P2F1F1P1P1P1
34	[illegible]	[illegible]	[illegible]	-1	2	P1P2P3F3F2P2P2P1
35	[illegible]	[illegible]	[illegible]	-1	1	P1P2P3F3F2S2P2P1
36	[illegible]	[illegible]	[illegible]	-1	1	P1P2P3S3F2S2P2P1
37	[illegible]	[illegible]	[illegible]	-1	1	P1P2P3S3F2P2P2P1
38	[illegible]	[illegible]	[illegible]	-1	2	P1P2P3F3P2P1P1P1