THE DEVELOPMENT OF MICROCOMPUTER SOFTWARE FOR THE PROCESSING AND INTERPRETATION OF SHALLOW REFLECTION SEISMIC DATA

Submitted to the Faculty of Science in partial fulfillment of the requirements for the M.Sc degree in Exploration Geophysics.

By Andrew Tertius Dippenaar

University of Pretoria

January 1988

THE DEVELOPMENT OF MICROCOMPUTER SOFTWARE FOR THE PROCESSING AND INTERPRETATION OF SHALLOW REFLECTION SEISMIC DATA

By : Andrew Tertius Dippenaar

Promotor : Prof. W. J. Botha

Department of Geology

Submitted for the M.Sc. degree in Exploration Geophysics.

Software was written to process shallow reflection seismic data on a Hewlett-Packard microcomputer. The technological advances made in the microcomputer industry in the last few years make it possible to process higher volume data on microcomputers than were possible a few years ago. Microcomputers are the ideal tools for plotting data, shifting plot axes, extracting data subsets and performing repetitive numerical calculations to enhance seismic data.

Programs were written and developed to restore seismic amplitude losses, correct for the near surface, attenuate frequency dependent noise, do velocity analysis, correct for source - geophone offset (normal moveout), stack common midpoint data and plot the final seismic section. A ray-tracing program was written to model simple geological models as well as a program to generate a synthetic common midpoint gather.

DIE ONTWIKKELING VAN MIKROREKENAARSAGTEWARE VIR DIE VERWERKING EN INTERPRETASIE VAN VLAK SEISMIESE REFLEKSIEDATA

Voorgelê ter vervulling van 'n deel van die vereiste
vir die graad
MSc in die Fakulteit Wis- en Natuurkunde.

deur Andrew Tertius Dippenaar

Universiteit van Pretoria

Januarie 1988

DIE ONTWIKKELING VAN MIKROREKENAARSAGTEWARE VIR DIE VERWERKING EN INTERPRETASIE VAN VLAK SEISMIESE REFLEKSIEDATA

Deur : Andrew Tertius Dippenaar

Promotor : Prof W.J. Botha

Departement Geologie

Ingedien vir die MSc graad in Eksplorasiegeofisika

Sagteware is ontwikkel om vlak seismiese refleksiedata op 'n Hewlett-Packard mikrorekenaar te verwerk. Die tegnologiese ontwikkelings wat onlangs op die gebied van mikrorekenaars plaasgevind het, maak dit deesdae moontlik om groter volumes data te verwerk met behulp van mikrorekenaars. Mikrorekenaars is die ideale hulpmiddel om data uit te stip, stipasse te skuif, datasubstelle te bekom en om groot volumes bewerkings mee te doen.

Programme is ontwikkel om amplitudemanipulasies te doen, om te korrigeer vir die effek van die vlakliggende aarde, om die effek van frekwensie-afhanklike ruis te verminder, snelheidsanalises te doen, te korrigeer vir bron - ontvanger-verplasing (normale uitbewegingskorreksie), gemeenskaplikemiddelpunt-data te stapel en om seismiese seksies uit te stip. Verder is daar ook 'n program geskryf om die stralepad wat 'n seismiese golf deur 'n eenvoudige geologiese model sal volg te bereken, sowel as 'n program om 'n sintetiese gemeenskaplikemiddelpunt-rekord te genereer.

THE DEVELOPMENT OF MICROCOMPUTER SOFTWARE FOR THE PROCESSING AND INTERPRETATION OF SHALLOW REFLECTION SEISMIC DATA

Submitted to the Faculty of Science in partial fulfillment of the requirements for the M.Sc degree in Exploration Geophysics.

By Andrew Tertius Dippenaar

University of Pretoria

January 1988

THE DEVELOPMENT OF MICROCOMPUTER SOFTWARE FOR THE PROCESSING AND INTERPRETATION OF SHALLOW REFLECTION SEISMIC DATA

By : Andrew Tertius Dippenaar

Promotor : Prof. W. J. Botha

Department of Geology

Submitted for the M.Sc. degree in Exploration Geophysics.

Software was written to process shallow reflection seismic data on a Hewlett-Packard microcomputer. The technological advances made in the microcomputer industry in the last few years make it possible to process higher volume data on microcomputers than were possible a few years ago. Microcomputers are the ideal tools for plotting data, shifting plot axes, extracting data subsets and performing repetitive numerical calculations to enhance seismic data.

Programs were written and developed to restore seismic amplitude losses, correct for the near surface, attenuate frequency dependent noise, do velocity analysis, correct for source - geophone offset (normal moveout), stack common midpoint data and plot the final seismic section. A ray-tracing program was written to model simple geological models as well as a program to generate a synthetic common midpoint gather.

DIE ONTWIKKELING VAN MIKROREKENAARSAGTEWARE VIR DIE VERWERKING EN INTERPRETASIE VAN VLAK SEISMIESE REFLEKSIEDATA

Deur : Andrew Tertius Dippenaar

Promotor : Prof W.J. Botha

Departement Geologie

Ingedien vir die MSc graad in Eksplorasiegeofisika

Sagteware is ontwikkel om vlak seismiese refleksiedata op 'n Hewlett-Packard mikrorekenaar te verwerk. Die tegnologiese ontwikkelings wat onlangs op die gebied van mikrorekenaars plaasgevind het, maak dit deesdae moontlik om groter volumes data te verwerk met behulp van mikrorekenaars. Mikrorekenaars is die ideale hulpmiddel om data uit te stip, stipasse te skuif, datasubstelle te bekom en om groot volumes bewerkings mee te doen.

Programme is ontwikkel om amplitudemanipulasies te doen, om te korrigeer vir die effek van die vlakliggende aarde, om die effek van frekwensie-afhanklike ruis te verminder, snelheidsanalises te doen, te korrigeer vir bron - ontvanger-verplasing (normale uitbewegingskorreksie), gemeenskaplikemiddelpunt-data te stapel en om seismiese seksies uit te stip. Verder is daar ook 'n program geskryf om die stralepad wat 'n seismiese golf deur 'n eenvoudige geologiese model sal volg te bereken, sowel as 'n program om 'n sintetiese gemeenskaplikemiddelpunt-rekord te genereer.

CONTENTS

LIST OF FIGURES

AKNOWLEDGEMENTS

The contribution of the following persons is hereby aknowledged :

* Brian Milner - for the idea of data processing on a microcomputer as a subject for a MSc thesis, the acquisition of the data used in the examples of this thesis and the comments on the earlier drafts of this thesis.

* Prof W. J. Botha - for the use of the HP microcomputers of the Geophysics section of the University of Pretoria as well as being my promotor.

* The energy fund of the C.S.I.R. for the funding of the seismic research at the University of Pretoria.

* SASOL (Pty) Ltd. - for funding the purchase of the seismic equipment at the University of Pretoria and the permission to use data from their coal mine areas.

* SOEKOR (Pty) Ltd. - for the use of their word processing facilities.

On a more personal level :

* My parents - For their support and the proof reading of many versions.

* Christine - For her support and all the help finalising this thesis.

1 INTRODUCTION

The analysis of shallow-reflection seismic data is very much an art. The desired seismic signals are hidden by random and non-random noise and further complicated by lateral inhomogeneities that are a distinct feature of the near surface earth. All these factors combine to make it very difficult for a computer to recognize reflection arrivals or produce a inverse model directly from the data without any human interaction. The technological advances in the field of microcomputers during the last few years make it the ideal tool for plotting data, shifting plot axes, extracting data subsets and performing repetitive numerical calculations to enhance the seismic data to such a level that the interpretation of the data can be done with a higher degree of confidence than was possible a few years ago. The interaction between computer and geophysicist, with the computer doing the repetitive work and plotting the data during any processing stage, and the geophysicist doing the evaluation of the processing processes and interpretation of the final graphs or sections can produce accurate and quick results.

2 EQUIPMENT AND DATA ACQUISITION

The computer programs listed in Appendix IV have been written and developed on the Hewlett-Packard 9816 microcomputers of the Geophysics section of the University of Pretoria. The minimum system requirement is HP Basic version 3.0 or higher. The programs listed in appendix IV were written for a computer system with 1.3 Mbyte memory. The data

was collected using a 24 channel Bison GeoPro field unit equipped with RS232C data communication ports. The data was recorded at SASOL Pty. Ltd.'s coal mines and is used with their permission. The spread consisted of 24 100 Hz high-frequency geophones with a geophone separation of 5 m. The seismic source was .5 kg Pentolite explosives buried in shotholes 3 m deep. The shot move up was equal to the geophone spacing which meant that every geophone position became a source position as well.

The data used in the examples in appendix III was recorded to 192 msec using a recording filter passband of 7-1000 Hz. With a record length of 192 msec the sampling interval is .2 msec which allows unaliased signal recording up to a frequency of 2500 Hz (the Nyquist frequency). The Bison GeoPro unit which were used can store up to 15 shot records in internal memory. After 12 to 15 shots have been recorded (depending on whether refraction data was shot or not) the data in internal memory is dumped to disc using the Hewlett-Packard computer in the field. During this time (+/- 30 minutes) the twelve back geophones are moved forward and the next shotholes are loaded. At the end of the day the massed dump files on disc are sorted into shot-geophone files. All subsequent processing are done on the shot-geophone files. A typical processing flow is set out in Appendix I.

Shot records, common offset sections or processed data can be displayed and/or printed at different output gains at any time. The following processes can be applied to the seismic data :

* amplitude recovery

* static corrections

* velocity analysis

* normal moveout corrections

* band-pass filtering

* final plots

* ray tracing

User instructions are listed in appendix II and program listings in appendix IV.

3 FACTORS THAT INFLUENCE SEISMIC AMPLITUDE

As a propagating wave moves away from its source, the energy associated with the wave's motion diminishes in amplitude. The rate of decay is quite rapid at early times becoming less pronounced at later times. O'Doherty and Anstey (1971) identified five major factors that contribute to seismic amplitude losses :

i. Spherical divergence

ii. Interface reflection coefficients

iii. Absorption

iv. Interface transmission losses

v. Multiple reflections

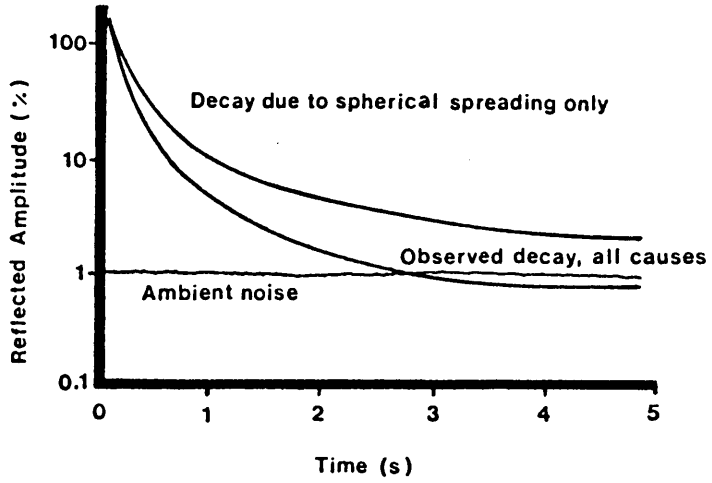


Figure 1: The decay of seismic amplitude

3.1 Spherical divergence

As a seismic wave propagates through a homogeneous constant-velocity medium, the wavefront has a perfect spherical shape. The energy associated with the motion of the wave is uniformly distributed over the spherical shell which the wavefront occupies at any instant of time. As the area of the expanding spherical shell increases with distance from the shot (area proportional to the square of the distance) the same amount of energy is being spread out over a larger

area resulting in an energy density decrease. Because amplitude is proportional to the square root of energy, it follows that amplitude should bear an inverse first power relation to distance from the source which in a constant velocity medium equates to an inverse first power relation to time.

3.2 The effect of reflection interfaces on amplitude

The presence of layering in the earth has a profound effect on the amplitude of the seismic signal. At every velocity interface some of the energy gets reflected back to the surface and some transmitted deeper into the earth. Four wave types are generated at reflecting interfaces - reflected P and S waves and transmitted P and S waves. The partitioning of energy between these wave types are expressed by the Zoepprits equations, the direction of propagation by Snell's law.

Using the notation of figure 2 Snell's law is :

$$\frac{\sin S_i}{W_1} = \frac{\sin P_i}{V_1} = \frac{\sin S_r}{W_1} = \frac{\sin P_r}{V_1} = \frac{\sin S_t}{W_2} = \frac{\sin P_t}{V_2}$$

with V the P-wave and W the S-wave velocities

See Waters (1981 : 45) or Sheriff and Geldardt, v. 1 (1982 : 65) for a summary of Zoeppritz's equations.

As a general rule velocity increases with depth. This leads to an outward bending of ray paths at velocity interfaces. This distorts the spherical wavefront and results in a more rapid decrease of

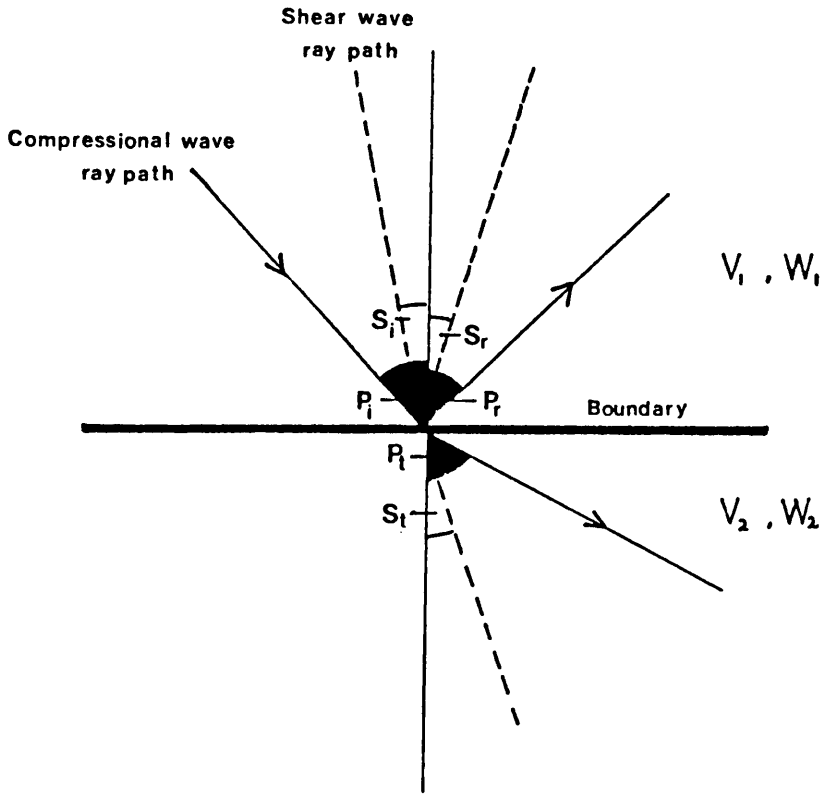


Figure 2 : Angular relations between incident, reflected and refracted rays at velocity interfaces

seismic amplitude.

The transmitted (refracted) wavefront has an amplitude in the lower medium that decays exponentially with depth below the boundary. The rate of decay depends on two factors :

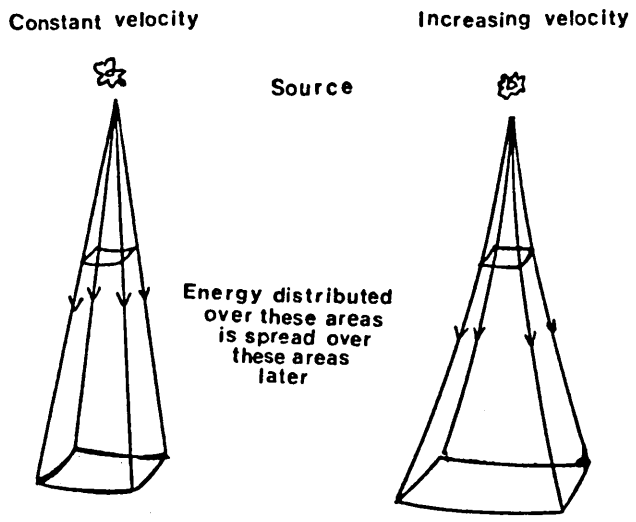


Figure 3: The expanding wavefront in a constant velocity and in an increasing velocity medium

i. the velocity of the lower medium (which governs the angle of refraction and thus the rate of spreading of the wavefront)

ii. the absorption characteristics of the lower layer.

3.3 Absorption

Although spherical divergence acts to diminish seismic amplitude with

distance, it does not involve any loss of seismic energy - merely a spreading of the energy over an expanding wavefront. The same argument applies to the effect of reflecting interfaces - no losses of energy - merely a redistribution of energy.

On the other hand, absorption diminishes seismic amplitude by an irreversible conversion into heat. Absorption is a function of the distance travelled and is ultimately responsible for the disappearence of the down-going wavefront. The mechanism by which energy is transformed into heat is not clearly understood. During the passage of a wave, heat is generated during the compressive phase and absorbed during the expansive phase. The process is not perfectly reversible since the heat conducted away during the compression is not equal to the heat flowing back during the expansion. (Telford 1976 : 241)

The quantification of absorption is not easy. Absorption coefficients can only be measured in a laboratory. Because laboratory measurements are invariably made at much higher frequencies than the frequencies found in the seismic spectrum there is no guarantee that the measured absorption coefficients apply to actual field data. This has lead to a wide divergence in absorption measurements.

The following factors have been recognized as contributing to absorption :

* Internal friction : The sliding of one rock grain against another along grain boundaries as a seismic wave propagates through the rock will cause friction. This

implies actual motion of one particle relative to another which is not easy to visualize in a well-sorted, well-cemented rock. In a poorly sorted highly fractured rock with a pattern of grain contacts of which some are in intense compression but others are not, particle motion relative to another particle can be visualized more readily.

* Imperfections in the structure of the material : small dislocations in crystal structure or minute pores with some sharp angular boundary can result in local perturbations in elasticity which will cause stress concentrations that will change stress-strain relations locally.

* Totally or partially filled pore spaces : this leads to dissipation of acoustic energy into heat energy. Experiments done on a wide variety of rock specimens at different humidity levels have shown that the adsorption of moisture on the grain boundaries has a profound effect on the absorption characteristics of the specimens.

In many physical phenomena the loss of energy by absorption is exponential with distance. This appears to be approximately true for elastic waves in rocks.

3.4 Other factors influencing amplitude

Many other factors have been identified that contribute to seismic amplitude loss. Some of them are :

* the energy involved in creating new surfaces (fracturing) near the source
* dispersion of energy as a result of the differences between phase and group velocity
* scattering of energy when reflecting off a point reflector
* energy losses attributed to the generation of multiples
* energy losses resulting from weak shots, bad geophone plants and near-surface anomalous conditions.

In more recent times the linear dependence of attenuation on frequency is being questioned. Jacobson (Jacobson 1987) is one of the scientists questioning this linear relationship. By measuring velocity dispersion (the difference between group and phase velocity), and using the Kramers-Kronig relation which exists between dispersion and attenuation, he attempts to distinquish between intrinsic and apparent attenuation (Apparent attenuation being the loss of seismic amplitude due to scattering of the wave). Jacobson concluded that there is no linear relationship between attenuation and the first power of frequency and that the attenuation coefficient cannot be

described by a single parameter, because it can not predict the amount of frequency-independent attenuation measured in a finite frequency band.

4 CORRECTIONS TO COMPENSATE FOR AMPLITUDE LOSSES

4.1 Spherical divergence correction

Because an inverse relationship exists between amplitude and time in a constant velocity medium, it follows that :

$$A_o t_o = A_n t_n$$

To bring amplitude A up to the reference amplitude A at time t simply implies multiplying A by t/t_o.

More sophisticated routines take into account the effect of velocity increasing with depth. The distance r from the source is expressed as velocity (V) time (t) and the correction calculated.

An example of such a routine is one developed by Newman (1973). He shows that for a horizontally layered earth and vertical ray paths the spherical divergence factor D can be expressed as :

$$D_o = tV^2/V_1 \quad \text{with} \quad V^2 = \Sigma t_i V_i / \Sigma t_i$$

This expression can also be extended to allow for non-vertical ray paths but this would imply doing ray tracing to determine the incident, transmitted and reflected angles before the correction can

be applied. The correction is a function of offset and angle of incidence which are not independent of one another but related by Snell's law. (Newman 1973).

To compensate for the exponential decay of amplitude due to spherical divergence as well as absorption, the following formula is sometimes used :

$$A = A * K * t^n * e^{aT} \qquad \text{with } T = t/t_o$$

Spherical divergence in itself contains no geological information.

4.2 Corrections for amplitude losses other than spherical divergence

Even with all the research that went into amplitude attenuation and related fields of wave propagation studies it is not a simple task to correct for amplitude loss caused by factors other than spherical spreading. Because there is such a wide variety of factors involved and because all the factors are not very well understood it is frequently necessary to resort to statistical processing as opposed to deterministic processing used in correcting for spherical spreading. In statistical processing the effect is measured on average and corrected on average.

Trace balancing is a form of statistical processing. Traces are balanced using a sliding-window averaging method. The mean amplitude of the values in every window position is calculated and the centre

amplitude is scaled to a reference amplitude using a scaler calculated from the calculated mean value. This process is also known as time-variant scaling or automatic gain control (AGC). However, an AGC-type gain can destroy signal character and must, therefore, be considerd with caution.

Traces are also usually normalized with respect to one another by calculating trace means, determining a scaler and scaling every amplitude on the trace to a reference amplitude. This process, called trace normalization, will boost low energy traces caused by weak shots and/or bad geophone plants.

5 NEAR-SURFACE TIME DELAYS

If delays in reflection time caused by the low-velocity near-surface layer are not corrected, entirely spurious structure can be introduced in the imaging of the subsurface. When doing high-resolution seismic profiling, where the aim is to identify faults with a throw of only a few metres, these corrections become very important. Because these corrections are applied to entire traces it is known as static corrections. Corrections where trace segments are shifted within traces are known as dynamic corrections (e.g. normal moveout corrections).

Near-surface time delays arise from two distinct causes :

i. elevation changes along the survey line

ii. strong velocity variations caused by lateral inhomogeneities in the near surface layer

The correction nescessary to shift an entire trace by the amount which would place events at two-way times that would have been obtained if the source and geophone positions were on a datum plane, with no weathering and low-velocity material present, is known as static corrections.

Static corrections can be calculated by several methods. Some of these methods include :

* using uphole data
* refraction first breaks
* event smoothing

5.1 Uphole data

Static corrections based on uphole data involve the direct measurement of vertical travel times from the buried source to a surface geophone next to the shot hole. In practice the geophone will not be directly over the shot hole but the offset should be small enough for the ray path of the direct wave to be regarded as vertical.

If the the shot moveup equals the geophone seperation every geophone

positon becomes a shot position as well which make static correction calculations based on uphole time measurements the ideal method to use. These type of static corrections are very accurate if good field acquisition practices were followed and should always be used when feasible (Sheriff & Geldart 1982). When doing shallow-seismic reflection work, the ray paths are very seldom vertical however and the uphole method will not allways give the best results.

5.2 Refraction first breaks

This method uses refractions off the bottom of the low-velocity layer to determine the velocity and thickness of the low-velocity layer. Corresponding time corrections are calculated from this data and the traces shifted by that amount. If the shot holes are not below the low-velocity layer or if the datum plane is a lot deeper than the bottom hole elevations a combination of the refraction and uphole methods can be used.

5.3 Event smoothing

Cross-correlating seismic traces and shifting them with regard to one another until the maximum correlation is obtained is one of the more sophisticated static correction routines. It is also known as event smoothing or automatic static corrections. The amount of shift allowed, as well as the correlation method used, varies from method to method.

6 CALCULATION OF STATIC CORRECTIONS

Every static correction consists of two parts :

i. a source static

ii. a receiver static

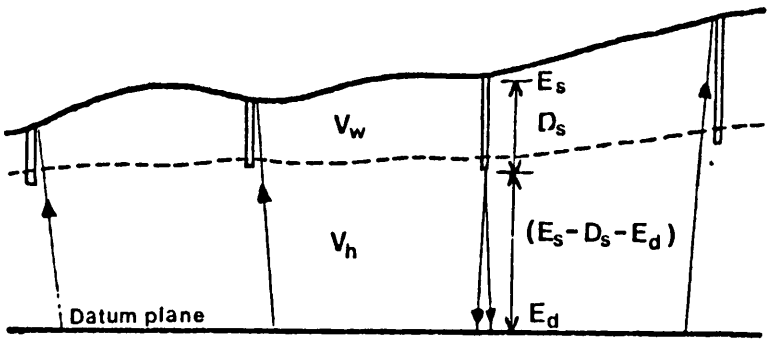


Figure 4 : Calculation of static corrections

with : T_o - shotpoint arrival time

E_d - datum plane elevation

E_s - surface elevation at the shot hole

D_s - depth of shot below surface

T_{uh} - uphole time measured

D_{ts} - source static

D_{tg} - geophone static

V_w - velocity of weathered layer

V_h - velocity of non-weathered layer

Source static : The source static is the time corresponding to the

distance travelled by the down-going wavefront from the source position to the datum plane. It is calculated as follows :

$$D_{ts} = (E_s - D_s - E_d) / V_h$$

Receiver static : The receiver static is the time corresponding to the distance travelled by the returning wavefront from the datum plane to the geophones on the surface. It is calculated as follows :

$$D_{tg} = D_{ts} + T_{uh}$$

The total static correction for a trace is the sum of the two equations and is expressed as :

$$D_{to} = D_{ts} + D_{tg} + T_{uh}$$
$$= 2((E_s - D_s - E_d) / V_h) + T_{uh}$$

Subtracting the total static (D_{to}) from every sample time is equivalent to placing the source and geophones on the datum plane, thereby eliminating the effect of the highly varying low-velocity layer and any effects due to elevation changes.

7 SEISMIC NOISE

A seismic trace consists of two elements : signal and noise. The term 'signal' denotes any event on the seismic record from which we wish to obtain information. Everything else is noise. The signal-to-noise ratio, abbreviated S/N, is the ratio of the signal energy in a specified portion of the record to the total noise energy

in the same portion.

Noise can be generated by artificial sources such as traffic. railroads, aircraft or natural sources like wind swaying bushes and grass, or the fluctuating pressures in the air caused by wind variation due to topography. There are two ways in which noise can be classified - either on the coherency of the noise or the repeatability of the noise.

Coherent noise is noise that can be followed across more than one trace. Examples of irrepeatable coherent noise include vehicular traffic, people walking near the spread or sporadic mining activity. Noise generated by surface waves is an example of repeatable coherent noise.

Incoherent noise is dissimilar on all traces and cannot be predicted on a trace from a knowledge of nearby traces. Incoherent noise is often referred to as random noise which not only implies unpredictability but also some statistical properties which are exploited in some of the processing routines. Examples of irrepeatable incoherent noise is energy scattered from near-surface irregularities and inhomogeneities such as boulders. This type of noise source is usually so small and so near the spread that the outputs of two geophones will only be the same if the two geophones are placed side by side. Non-repeatable random noise may be due to wind shaking a poorly planted geophone or caused by the roots of trees moving due to strong winds or by stones ejected from a poorly tamped shot and falling back to the earth near or on a geophone.

8 ATTENUATION OF NOISE

The signal-to-noise ratio generally depends on frequency. (Sheriff & Geldart 1984 : 126). If the noise has appreciable energy outside the principal frequency range of the signal, frequency filtering can be used to advantage. Frequency is of limited value in improving record quality if the frequency spectrum of the noise overlaps the signal spectrum.

8.1 Frequency filtering

Frequency filtering is the attenuation of certain components of a signal based on frequency. It may be done by analogue (electrically within the recording unit) methods or numerically (digital filtering). Digital filtering have many advantages over analogue filtering :

* no thermal variations in components
* eliminaton of periodically calibration
* repeatability and stability of performance
* programmability for design flexibility
* precision is only limited by computational word length

8.1.1 Frequency and time domain concepts

Seismic data is recorded in the time domain. If an extended analysis

of the signal is required it is useful to transform the signal to the frequency domain. The frequency-domain method consists of building up a series of sinusoidal signals of known peak amplitude and phases. The frequencies are related to a fundamental frequency in a harmonic manner which means that the sinusoids are all integral multiples of this fundamental frequency. Alteration of the amplitude and phases of the constituent sinusoids in the frequency domain produces a different time domain representation, and this can be regarded as a fundamental definition of filtering.

Because frequency filters are specified in the frequency domain, it is necessary to do either a Fourier analysis of the seismic data to convert it into the frequency domain or a Fourier synthesis of the filter to convert it into the time domain. A filter is fully characterized by its transfer function. The transfer function is represented by amplitude-vs-frequency and phase-vs-frequency curves (amplitude and phase spectra). The time-domain representation of the transfer function is called the impulse response of the filter. If the filtering is carried out in the time domain the traces to be filtered are convolved with the impulse response of the filter. If the filtering is carried out in the frequency domain the amplitude spectra of the filter and data are multiplied together while the phase spectra are added together.

The phase shift of a filter is that time (expressed as a fraction of the wave period) that every sinusoidal frequency component must be shifted to bring its zero crossing to zero on the time axis. A

negative time shift corresponds to a positive phase value. If a filter is not zero phase, the wave shape of the pulses (which is the sum of the sinusoidal Fourier components) will be altered. Ideally a filter should be zero-phase. A zero-phase filter is symmetrical about time zero.

Frequency filters are classified as low-pass, high-pass, band-pass or band-reject (notch) filters depending on the way the filter discriminate against frequencies above or below a certain limiting frequency or outside of a given band of frequencies. The ideal filter is a filter that would leave fequencies within the passband unaffected and cut out all frequencies outside the pass band. Mathematically these ideal filters can be expressed as :

$$\text{Ideal low-pass filter : } F(w) = 1 \quad |w| < |w_2|$$
$$= 0 \quad |w| > |w_2|$$

$$\text{Ideal high-pass filter : } F(w) = 0 \quad |w| < |w_1|$$
$$= 1 \quad |w| > |w_1|$$

$$\text{Ideal band-pass filter : } F(w) = 1 \quad |w_1| < |w| > |w_2|$$
$$= 0 \quad |w| < |w_1| \text{ or } |w| > |w_2|$$

with w_1 and w_2 the cutoff frequencies

The amplitude spectra and impulse responses of ideal filters are shown in figure 5.

There are problems associated with these ideal filters. To achieve

vertical cutoff slopes an infinite impulse length response is needed. The effect of using finite impulse responses is to introduce ripples within, as well as outside, the pass band - the effect being noticed especially near the cut-off frequencies - this is the well-known Gibb's phenomenon. The way to reduce the ripple effect is to introduce sloping cut-off sides at the cut-off frequencies. The more gentle the slope the less ringing (ripples) occurs but the less effective the filter.

8.1.2 Recursive filtering

Filtering does involve a lot of repetitive calculations and are thus computer intensive. If filtering is done by convolution each output point is the weighted sum of a finite number of input points. If a recursive algorithm is used the output points are the weighted sum of input points as well as previously computed output points. The advantage of using recursion is that some filtering operations can be done a lot faster than the same filtering operation using convolution (Shanks 1967).

Recursion filters must be stable and respond only in positive time. Therefore in general they do not have linear or zero phase spectra. If the input time series is finite in length, simple manipulation produces a filtering operation with linear phase response. To produce zero phase recursion filters, the input is filtered recursively with the designed filter. The output from this first filtering operation is then reversed in time and filtered with the same recursive filter.

The output of this second pass is again reversed to give the filtered time series. The phase response of this filtering operation is zero for all frequencies, while the amplitude spectrum is the square of the filter's original amplitude spectrum. For band-pass filters this presents no problem as the squaring of the ampliude spectrum produces sharper cutoff slopes. (Shanks 1967)

When applying high-order recursion filters care must be taken that round-off errors in the coefficients do not cause instability. To avoid this problem cascaded filters can be used.

8.2 Common midpoint Stacking

Random noise is best attenuated using common midpoint (CMP) stacking.

CMP stacking entails the gathering of traces that share a common midpoint between shot and geophone, determining the stacking velocity function that will lead to the best stack at that midpoint, correcting the gather for the normal moveout associated with that velocity, muting the stretched part of the gather and stacking the gather. The result of this process is the stacking (emphasizing) of all reflectors that line up and the attenuation of random events (noise) on the gather that does not line up.

The variation in reflection arrival time because of shot point to geophone distance (offset) is defined as the normal move-out (NMO) of the reflector. The additional time required for the energy to travel from the source to a reflector and back to an offset geophone as

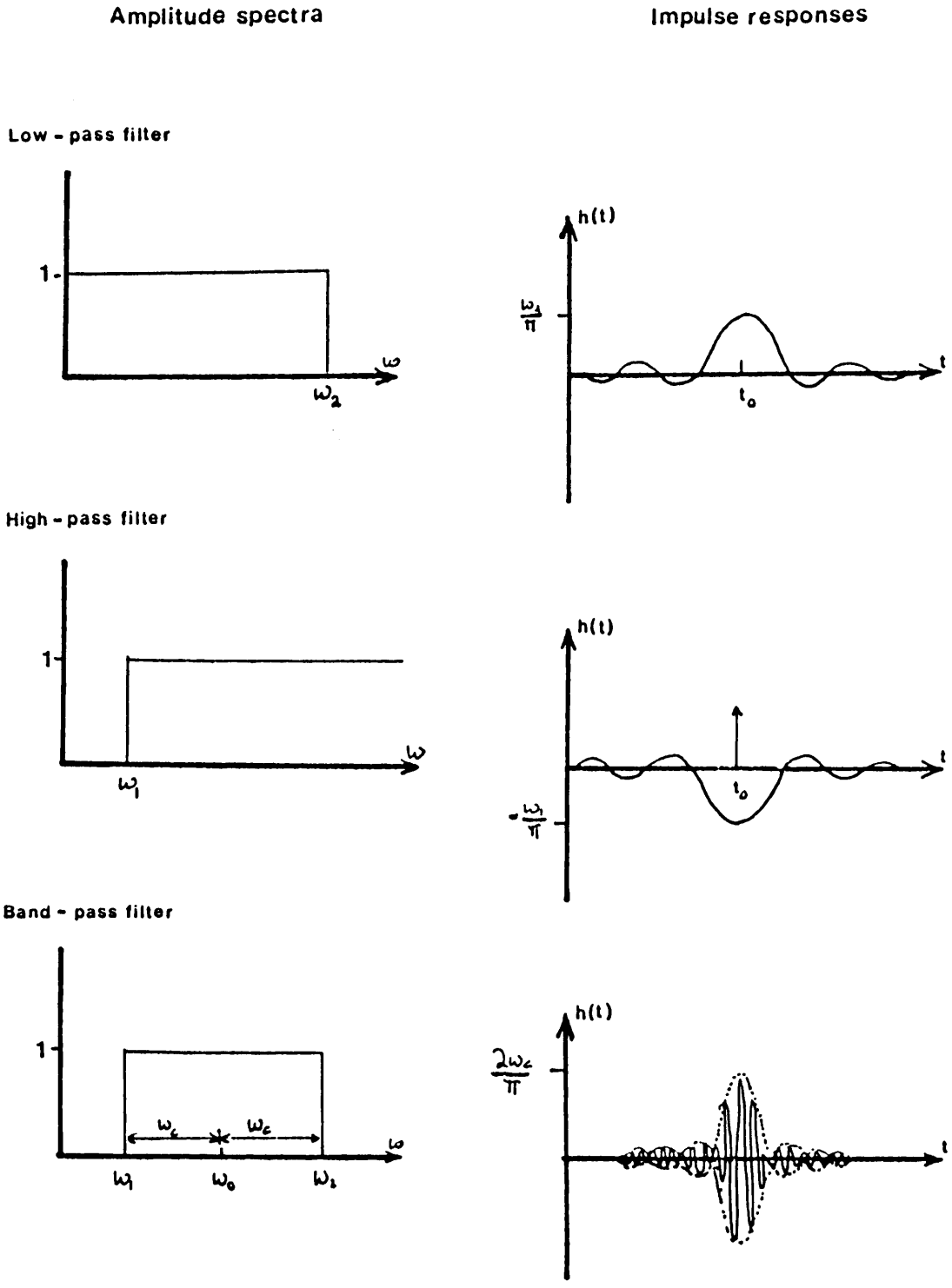


Figure 5 : Amplitude spectra and impulse responses of ideal filters

compared to a zero offset geophone is called the normal move-out correction.

The normal move-out functions used to correct CMP (common midpoint) gathers prior to stacking are hyperbolic functions defined by :

$$T^2 = T_o^2 + X^2/V^2$$

with : T_o - vertical two-way time

T - two-way time to offset geophone

X - offset from source to receiver

V - stacking velocity

Care must be taken when using CMP stacking with high-resolution work as an inaccurate velocity function can lead to a wrong interpretation of the shape (structure) of a reflector. When the subsurface is not horizontal the energy is reflected off different points in the subsurface and the stacked trace will not be an true representation of the subsurface.

8.3 The advantages of CMP stacking

Some of the advantages of CMP stacking are :

i. Attenuation of noise : S/N = SQRT(fold of cover)

* CMP traces are recorded at different times and thus different ambient noise levels.

* source generated noise e.g. ground roll and the passing air wave is attenuated.

ii. Subsurface continuity :- Single fold coverage is vulnerable to missed shots, bad geophone plants and instrument malfunctions. CMP stacking overcomes this problem by providing several ray paths that sample the same point (portion) of the subsurface.

iii. The effect of near-surface anomalous low-velocity areas in the vicinity of a source or geophone position is reduced.

iv. "Blind spots" caused by the sides of subsurface bodies refracting the seismic wave away from the geophone position are no longer possible.

9 VELOCITY ANALYSIS

Determinating the NMO correction of a CMP gather is equivalent to determining the velocity structure of the earth at that particular CMP

gather point. This stacking velocity or NMO velocity is the velocity of a constant homogeneous isotropic layer above a reflector which would give approximately the same offset-dependence (normal moveout) as actually observed. In the limit as offset approaches zero, it is the same as the root-mean square (rms) velocity V_{rms}. The root-mean square velocity refers to a specific raypath and is defined as :

$$V_{rms} = [\Sigma(V^2_i t_i)/\Sigma t_i]^{1/2}$$

Two methods of velocity analysis have been included into these processing routines and are the only two methods that will be discussed.

9.1 Constant-velocity gathers

Using a constant velocity over the entire trace, a normal moveout value is calculated for every sample point on the trace. The amplitude at the sample point correspondir, to the calculated normal moveout, is moved to its new T_o sample position. If the calculated normal moveout position falls between two sample positions, an interpolated value is used. The whole CMP gather is corrected in this manner for a set of different velocities and displayed. The velocity that lines up a reflecting event the best will give the best stacked result for that particular event. An event that curves upwards in a NMO corrected CMP gather is undercorrected - the true stacking velocity to the reflector is less than the velocity used to correct the gather. By the same token - an event that curves downwards toward

the outer traces is overcorrected. Note that higher velocities gives less moveout to events. By determining what velocity lines each identifable reflector up the best across the gather, a velocity function (velocity vs. two way time) can be picked that will best stack up the CMP gather. If the S/N ratio is very low it may be difficult to line up events to select this stacking velocity function.

9.2 Constant-velocity stack

The procedure for constant velocity stack (CVS) analysis is the same as that of the constant velocity gather (CVG) method- i.e. calculating NMO using constant velocities and applying the calculated NMO to the traces. The NMO corrected gather is then stacked to form a single CMP stacked trace. A few adjacent CMP gathers are also stacked with this constant velocity. The whole procedure is repeated for a number of different velocities. The different constant-velocity panels are then displayed. Where the NMO has been correctly removed the events will stack up to produce a sharp, strong event. The problem with the CVS method is picking the best stacking velocities from a few panels where no appreciable differences can be spotted. Quantitive interpolation between different velocity panels is difficult.

Whenever feasible, both the CVG as well as CVS method should be used. As the final result is to be a stacked section when velocity analysis is done, it makes sense to use the CVS method. However, the CVG method is of extreme importance if a more accurate velocity estimation

is to be done.

9.3 Some considerations on normal moveout.

Large normal moveout values on the outer traces can lead to anomalous low frequencies in the upper part of a CMP gather. To prevent the degrading of the stack by this induced low frequency, the outer traces are muted - a process known as stretch muting. By doing this however, the stacked trace are not balanced anymore as more zeros are included in the upper part of the gather. To compensate for this loss of fold in the upper part of the stack, the stacked trace is normalized by dividing each sample's amplitude by the square root of the fold of the stack at that point on the trace.

10 CONCLUSIONS

As with every other geophysical technique the secret to good results lies with good field techniques and data acquisition practices - not just the seismic data itself but also the data needed to calculate the static corrections. Data processing can enhance the data to produce excellent seismic sections if the field data is good. Data processing can not turn bad field data into good field data. The processing of seismic data involves a lot of calculations and manipulation and is by nature time consuming. Therefore it make sense not to waste this time trying to salvage bad field data but rather to enhance good field data.

It is up to the user of these programs to decide whether the improvement in data quality (if any) justifies the extra time involved in doing velocity analysis, correcting for normal moveout and stacking the common midpoint data as opposed to scaling and filtering a constant offset section - the offset so selected as to reflect the events of interest the best. From the data available to test these programs I do not feel that the extra time is justified. Theoretically the improvement in S/N with six fold CMP stacking is a factor of 2.4, but the lack of identifable reflectors deeper down the traces made sensible velocity analysis impossible.

Notwithstanding this, it has been demonstrated that processing of seismic data is possible on microcomputers. Until more efficient input and output mechanisms can be incorporated into microcomputers, the applications of processing routines on microcomputers will be

limited to the relative low volume data of shallow-reflection seismic work. As arithmetic processors for microcomputers are becoming more readily available, routines like deconvolution which involves a lot of calculations can also be adapted to run on microcomputers.

11 REFERENCES

ANSTEY, N.A. et al. 1985. **IHRDC Video library for exploration and production specialists : GP304 - Multiple coverage.** Boston : International Human Resources Development Corporation

ANSTEY, N.A. et al. 1985. **IHRDC Video library for exploration and production specialists : GP401 - Basic processing.** Boston : International Human Resources Development Corporation

CLAERBOUT, J.F. 1976. **Fundamentals of geophysical data processing: With applications to petroleum prospecting.** New York : Mcgraw-Hill Book Company

DOBRIN, M.B. 1976. **Introduction to geophysical prospecting. 3rd edition.** New York : Mcgraw-Hill Book Company

FATTI, J. 1974. **Digital processing of seismic reflection records from South Park Colorado.** Ph.D thesis, Colorado School of Mines, Golden Colorado.

HATTINGH, M. 1986. **Some basic and new ideas on digital filtering methods for geophysical data processing.** South African Geophysical Association course notes : Geophysical data processing.

HOFFMAN, J.L. 1986. **High resolution analysis of shallow seismic data.** New Jersey Geological Survey open-file report 86-1.

HUBRAL, P. & KREY, T. 1980. **Interval velocities from seismic reflection time measurements.** Tulsa, OK. : Society of Exploration

Geophysicists

JACOBSON, R.S. 1987. **An investigation into the fundamental relationship between attenuation, phase dispersion, and frequency using seismic refraction profiles over sedimentary structures.** Geophysics 52 : 72.

MCQUILLIN, R., BACON, M. & BARCLAY, W. 1985. **An Introduction to seismic interpretation.** London : Graham and Trotham.

NEWMAN, P. 1973. **Divergence effects in a layered earth.** Geophysics 38 : 481 Tulsa, OK.

O'DOHERTY, R.F. & ANSTEY, N.A. 1971. **Reflections on amplitudes.** Geophysical prospecting 19 : 430. The Hague.

PAPOULIS, A. 1962. **The Fourier integral and its applications.** New York : Mcgraw-Hill Book Company.

SHANKS, J.L. 1967. **Recursion filters for digital processing.** Geophysics 32 : 35. Tulsa, OK.

SHERIFF, R.E. & GELDART, L.P. 1982. **Exploration seismology - Vol 1 : History and data acquisition.** Cambridge : Cambridge University Press.

SHERIFF, R.E. & GELDART, L.P. 1983. **Exploration Seismology - Vol 2 : Data processing and interpretation.** Cambridge : Cambridge University Press.

SHERIFF, R.E. et al. 1984. **Encyclopedic dictionary of exploration**

geophysics 2nd edition. Tulsa, OK : Society of Exploration Geophysicists.

TELFORD, W.M. et al. 1976. **Applied geophysics.** Cambridge : Cambridge University Press.

WATERS, K.H. 1978. **Reflection seismology.** New York : Wiley & Sons

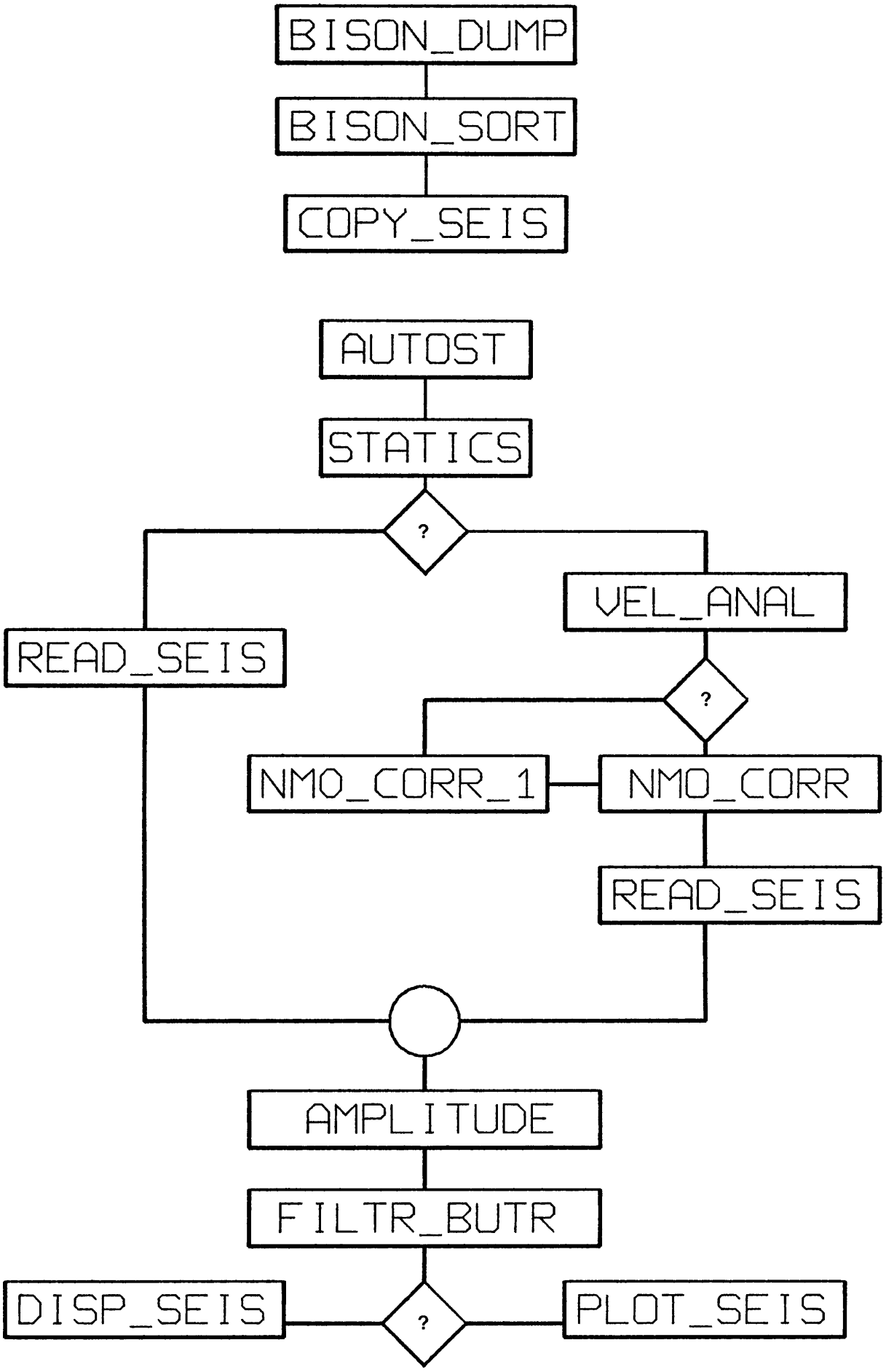
YILMAZ, O. 1987. **Seismic data processing.** Tulsa, OK : Society of Exploration Geophysicists.

ZIOLKOWSKI, A. & LERWILL, W.E. 1979. **A simple approach to high resolution seismic profiling for coal.** Geophysical prospecting 26 : 360

APPENDIX I

TYPICAL PROCESSING FLOW

PROCESSING FLOW



PROCESSING FLOW

The flowchart on the previous page shows suggested processing routes when using these programs. Depending on the data quality two routes can be followed :

1. Static corrections, amplitude restoration and filtering

2. Velocity analysis, NMO correction, stack, amplitude restoration and filtering.

For both these routes the very first step is to arrange the field data into shotpoint - geophone files using programs BISON_DUMP and BISON_SORT. If so desired the data can be copied to the hard disc using COPY_SEIS. The data is now in the correct file format for the processing routines.

Using program AUTOST the different processing routines can now be called. Using program STATICS the source and geophone static correction is calculated for every peg position on the line. Using READ_SEIS and DISP_SEIS to look at the data, a decision must now be made as to what processing route (as listed above) is going to be followed.

If the data is of sufficient quality, a constant offset section is read in (READ_SEIS), the amplitude restored and traces normalized (AMPLITUDE) and the data filtered if nescessary (FILTR_BUTR).

If reflectors are identifiable to assist velocity picking on the constant velocity gathers the more intensive CMP stacking route can be followed. Program VEL_ANAL is the velocity analysis program. After a velocity function has been picked, NMO_CORR_1 is used to determine the

stretch mute that must be applied to the traces prior to stacking. If satisfied with the result the rest of the line is stacked using NMO_CORR. The output from this two routines are stored on disc. Using READ_SEIS the data is read from disc, normalized (AMPLITUDE) and filtered (FILTR_BUTR) if nescessary.

With both these options the data is either displayed on the screen and dumped to a printer (DISP_SEIS) or plotted on a HPGL plotter (PLOT_SEIS) after processing.

APPENDIX II

PROGRAM DESCRIPTIONS AND USER INSTRUCTIONS

II.1 GENERAL INFORMATION

II.1.1 Filenames

The different filenames created and accessed by the different programs are :

A_LL_SSSM : Mass field records

LL_SSS_GGG : Unprocessed field data

LLnSSSnGGG : Unprocessed field data (negative positions)

LL_surface : The elevation and depth of shotholes on the line

LL_geo_st : The calculated geophone statics

LL_srce_st : The calculated source statics

SSSvVVgGGG : Constant velocity gathers

VVVVcvsMMM : Constant velocity stacked CMP gathers

LL_SSSfGGG : Filtered traces

SSSnmoGGG : NMO corrected CMP gathers

LL_MMM : Stacked section

with

A	area identification
LL	line identification
SSS	shotpoint number
GGG	geophone position
VV	constant velocity/100 used in constant velocity gather
VVVV	constant velocity used in constant velocity stacks
MMM	midpoint of CMP

II.1.2 The common memory block

All of the programs, with the exception of PLOT_SEIS, STATICS and RAY_TRACE share a common memory block in which variables that are 'shared' by the different routines reside. Whenever the above mentioned programs are executed these common variables are lost. The advantages of this common memory lies in the ease in which data can be displayed and compared. The input field data is in integer format whereas the processing of the data is done in real format. Program AMPLITUDE is the only routine that transcribes the processed data back to integer format before returning the user to the main menu. In program DISP_SEIS which is used to plot the traces to screen, the user has the option to plot either the integer or real format data arrays to screen. This is very useful when different tests are done as the input data is not destroyed during the processing and/or displaying of the data. Programs FILTR_BUTR, DISP_SEIS and AMP_SPECTR benefits a lot from this as it requires extensive testing to determine which band-pass filter enhances the data the best.

II.1.3 Variables used in the common memory blocks

COM block /Seis_int/

Trace(24,960) : Twenty four traces with 960 sample values in each trace (INTEGER format).

Opt : The type of data being processed/displayed (See READ_SEIS for descriptions of different options).

Amt : The number of traces being manipulated.

Gain(24) : The gain (in dB) at which every trace have been recorded.

COM block /Seis_real/

Shot(24) : Shotpoint numbers (actual ground position) of the traces being manipulated.

Geo(24) : Geophone positions (actual ground position) of the traces being manipulated.

Shift_value(24) : The number of samples corresponding to a calculated time value (e.g. with a sampling interval of .2 msec a time value of 23 msec will be equal to 115 sample points)

Offset : The offset used in a common offset section.

COM block /Processed/

Processed_trace(24,959) : Temporary REAL format array into which data is written while being processed.

Pointy : The position of the "menu arrow"

V : The velocity used to do constant velocity analysis (either CVG or CVS)

COM block /Polar/

Pol_factor(24) : Integer format array - the values of the array is either 1 or -1 depending on whether polarity is reversed or normal.

COM block /Statics/

Geophone_static(100) : Real format array - static correction (in msec) calculated to shift the trace by that amount which would place the geophone on the selected datum level

Source_static(100) : Real format array - static correction (in msec) calculated to shift the trace by that amount which would place the source on the selected datum level

Peg(100) : Real format array - true ground height at the peg positions

COM block /Screens/

Screen_1(7500) : Variable used to store graphics screen -- used when switching between programs that use the same display.

Screen_2(7500) : If more than 24 traces are displayed the display of the higher numbered traces are stored in this array.

X : Number of graphic display units along the horizontal axes of the graphics screen (X = 133 on the HP9816).

Y : Number of graphic display units along the vertical axes of the graphics screen (Y = 100 on the HP9816).

Gain_amplitude : The factor by which the amplitudes of the traces are adjusted prior to displaying the traces.

Gain_time : The factor by which the traces are stretched prior to displaying the traces.

Some more common variables (not in common blocks)

Sweep : The length of traces in msec (NB - A sweep length of 192 msec are assumed in all the programs. If another sweep length is used when recording the data, this variable must be changed accordingly in all the programs in which it occurs).

Sample_rate : Defined as Sweep/959 (There are 959 samples in every trace)

Sample_inverse : Defined as 1/Sample_rate.

In the discussion of the programs that follow variable names that are self explanatory are not listed. Where some confusion may arise the function of the variable is explained.

II.1.4 AUTOST

This is the main calling program for the different programs that are described in the following sections. The different processes are listed in menu's and is selected by moving a pointer along the side of the menu. If the screen dump options are selected the program checks whether the data has been written into the graphic arrays. If there is no data in the graphic arrays the user is returned to the main menu. The program DISP_SEIS should then be selected to draw the display prior to dumping the screen to the graphics printer.

Variables used in AUTOST :

Stack_ques$: Either "Y" or "N" - controls whether NMO_CORR or NMO_CORR_1 is called.

II.2 DATA TRANSFER AND INPUT/OUTPUT PROGRAMS

II.2.1 BISON_DUMP

This program controls the data transfer from the Bison Geopro field unit to the HP9816 microcomputer in the field. The data is sent in 16 bit format and is read from the enhanced memory of the Geopro unit. To transfer the data from the Geopro to the computer the following procedure must be followed :

1. Recall the field record that must be dumped from the Geopro's internal memory.

2. Prepare the Geopro unit for data transfer by pressing <WRITE> <7> <ENTER> on the Geopro. A message should appear on the Geopro's screen informing you "RS232 ready". If this message does not appear on the screen press <WRITE> <7> <ENTER> again.

3. As soon as the Geopro is ready (displaying "RS232 ready") press <CONTINUE> on the computer. The message should change to "RS232 active".

When all the data has been transferred to the computer (+/- 2 min) it is stored on disc.

Variables used in BISON_DUMP :

Area$: Area identification

Mixed_array(23200) : Integer format array - the sample values are dumped to this array prior to writing them to disc. The data is in trace consecutive order -- it consists of 24 traces each with 959 values followed by eleven trace paramaters.

II.2.2 BISON_SORT

This program reads the "mass" output files created by BISON_DUMP, extracts the 24 traces of the field record from the data and writes it to disc as 24 shotpoint - geophone files. All subsequent processing is done on the shotpoint - geophone files. There is an option to display the data prior to writing it to disc.

II.2.3 COPY_SEIS

This program copies sorted files from floppy disc to a hard disc (if so desired by the user). The data is stored on sectors two and three of the hard disc.

II.2.4 READ_SEIS

This program reads unprocessed or processed data from disc. There are seven different data sets accesible to READ_SEIS :

i. field records

ii. raw CMP gathers

iii. common offset sections

iv. constant velocity corrected CMP gathers

v. constant velocity corrected CMP stack panels

vi. corrected CMP gathers (corrected with a velocity function)

vii. final processed sections

The data that is to be read in is selected in the main menu program AUTOST. If unprocessed traces are read in (options 1, 2, 3) the user has the option to read in calculated static corrections.

Variables used in READ_SEIS :

Stat_ques$: Either "Y" or "N" -- determines whether static corrections should be read in or not.

Format$: Either "I" (integer) or "R" (real) -- Format in which the data have been stored in.

Cdp_centre : Midpoint (ground position) of the CMP gathers (options 2, 4, 6).

First_cdp : First midpoint of CMP's used in the constant velocity stack panels (option 5).

Mid : CMP midpoints for consecutive gathers used in constant velocity stack panels.

Position : Actual ground position of final stacked trace.

Dummy(959) : Integer format array - array into which every trace is read prior to being sorted into array Trace(*) (saves disc access time).

II.2.5 DISP_SEIS

This program displays the seismic traces that is stored in either the /Seis_int/ or the /Processed/ common memory blocks. The trace data can be manipulated in the following ways prior to displaying it :

i. stretched (expanded along the time axes).

ii. amplitudes can be adjusted in 6 dB (factor 2) steps.

iii. variable area (fill) or wiggle trace only can be selected.

If static corrections have been calculated and is present in the /Static/ common memory block the traces are shifted accordingly. The overlap between traces is null unless the amplitudes is adjusted. If this is done the amount of overlap will depend on the output gain

selected.

Variables used in DISP_SEIS :

Label$[80] : the label writen on the top of every display (depending on the option selected)

Var_area$: Either "ON" or "OFF" -- choice whether variable area (fill) or wiggle trace only display mode is selected.

Qq$: "I" or "R" -- same as Format$ in READ_SEIS

Screen_toggle : 0 or 1 -- toggles between Screen_1(*) or Screen_2(*) (NB .... Screen_2(*) not used if less than 25 traces)

II.2.6 PLOT_SEIS

This program has the same features as DISP_SEIS. It plots either a final stacked section or a common offset (unprocessed) section on a HPGL plotter. The plotted section can be done on A3 paper or on A1 paper halved along its long axes (called "A3 long" in the program). It is important to note that this progran does not share the COM memory blocks and that the data in the COM memory blocks will be destroyed when this program is executed. Because no unnecesarry variables are maintained it is possible to read in up to 100 traces. The program has its own reading routine which is similar to the one used in READ_SEIS. This program is usually executed on its own (not

called from AUTOST) or right at the end of the processing sequence after the data have been stored on disc.

II.3 PROCESSING PROGRAMS

II.3.1 STATICS

This program calculates source and geophone statics using uphole times. The line information (peg positions, peg spacing, peg elevation and depth of shot holes) are entered. The datum is selected as the lowest elevation of the bottoms of the shotholes. Ziolkowski and Lerwill (1979) have emphasized the need for a datum plane close to the average bottom hole elevation. The relevant zero offset section is read in and displayed on the screen. Using the ECHO the first breaks are picked on the screen. These uphole times are then used to calculate source and geophone statics. The calculated source and geophone statics as well as the elevations and the depths of the shot holes are stored on disc.

Variables used in STATICS :

Trace(24,960) : Integer format array - the zero offset section is stored in this array.

Upper_velocity : The calculated velocity of the zone between surface and datum level.

Uphole_time : The picked first breaks on the zero offset section

Height : The elevation of the peg positions along the line

II.3.2 POLAR

This program changes the polarity of selected traces. The program changes only the polarity on the displayed section. All processing is done on field polarity maintained traces.

Variables used in POLAR :

Pol_ch(24) : Integer format array - trace numbers of channels of which the polarity must be changed.

II.3.3 AMPLITUDE

This program restores amplitude losses that are attributed to spherical spreading as well as correcting amplitudes affected by weak shots or bad geophone plants by trace normalization or applying an AGC to the data. The data is manipulated in real format (in array Processed_trace(*)) and written into integer format (array Trace(*)) at the end of the program. There are four options available :

1. mute first breaks

2. correct for spherical spreading

3. trace normalization (between traces)

4. trace balancing (within traces)

Mute : If the first part of a trace is to be muted (e.g. the first breaks) a mute taper is specified. The start of the taper as well as the length of the taper is specified in msec. All sample values preceding the start of the taper is replaced by zeros and the sample values in the taper zone are tapered from zero to full sample value over the length of the tapered zone. If no muting is desired, the user is prompted for a time value (TO) at which the spherical divergence and/or AGC calculations will be started.

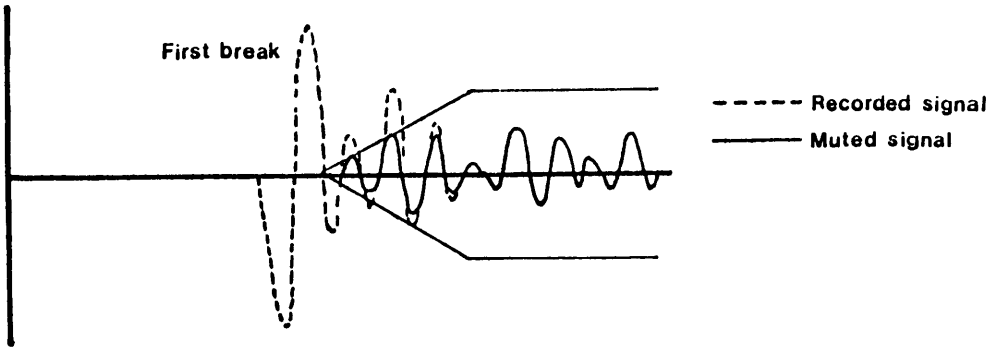


Figure 6 : Muting the first break

Spherical correction to traces : Two options are available - linear or exponential correction. The spherical correction is applied from either the end of the mute taper or from time TO. If the linear option is selected each sample is multiplied by its corresponding

two-way time divided by a reference time. If the exponential option is selected, the formula given in the paragraph on amplitude manipulation is used.

Normalizing traces : If it is necessary to adjust the overall amplitude of a trace, the trace is normalized by calculating the mean amplitude of the trace and computing a scaling factor to normalize this mean amplitude to a reference amplitude. Each sample value is then adjusted with this scaler.

Balancing a trace : To correct for factors influencing seismic amplitude other than spherical spreading a trace can be "balanced" using a sliding window down the trace, calculating the mean sample values in the window and scaling the centre sample value to a reference amplitude using the same procedure described in normalizing traces. The window is then moved up one sample position and the whole procedure repeated. The following points should be considered when deciding on the length of the sliding (averaging) window :

- the longer the window - the more equalized the amplitudes will be.
- the shorter the window - the less equalized the amplitudes will be. This is also the more computer intensive option of the two.

It is important to note that relative amplitudes are maintained - the weaker amplitudes become stronger and the stronger ones are reduced but a weak event will not suddenly become a strong event. The sample

values preceding the sample value of the midpoint of the first window are adjusted with the first calculated scaling factor, the sample values following the midpoint of the last window are adjusted by the last calculated scaling factor.

Variables used in AMPLITUDE :

Mute_ramp : Length of mute ramp in msec. (input)

Ramp_length : Corresponding length in number of samples. (calculated)

Mute : Start of mute for every trace in msec. (input)

Start_ramp(24) : Corresponding sample number at which mute taper start. (calculated)

End_mute(24) : The sample number at which the tapered zone stops.

Ramp_factor : Factor (0 to 1) with which current value are multiplied in ramp zone to adjust values from 0 to full sample value.

Trace_scaler : Calculated scaling factor with which every sample value are adjusted to normalize traces.

Scaler : Calculated scaling factor with which every midpoint of the sliding window is adjusted to balance a trace.

Reference_amplitude : Amplitude to which sample values are scaled to. (defined in the program - if another reference amplitude is desired the program must be edited)

Index : The sample number of the current midpoint of the sliding window

II.3.4 VEL_ANAL

This program does constant velocity analysis and outputs one of two options :

i. a constant velocity gather

ii. a constant velocity stacked panel

The program has its own reading routine to gather the CMP traces. If static data have been calculated for the line being processed the static data is read from disc and the static correction applied prior to doing the velocity analysis.

The NMO corrections are calculated using the hyperbolic moveout function described in the paragraph dealing with normal moveout. A moveout value is calculated for every sample value on the trace and stored in an array T(*). Before calculating the normal moveout the program checks to see whether the moveout data for the specified velocities have not ben generated during a previous velocity analysis. If the moveout data is available on disc the NMO values are read into

array T(*) - if not the data is stored on disc after calculating the moveout data.

Every sample value on the trace is shifted by the number of samples corresponding to the calculated normal moveout associated with that particular sample number for the given velocity and trace offset. If the calculated normal moveout lies between two samples (e.g. NMO is not a multiple of sample increment) linear interpolation between the adjacent samples is used to determine the value that must be shifted to the new sample position. The corrected trace is stored in array Cvg(*) as well as added to array Cvs(*).

If the CVG output option was chosen array Cvg(*) is stored on disc at this stage. The whole procedure is repeated until all the traces in the CMP gather have been corrected. If the CVS option was chosen the final stacked array (Cvs(*)) is stored at this stage and the whole procedure repeated until all the midpoints have been stacked.

The zero offset trace can be ignored in the stacking process if so desired. The reason for this lies in the inefficiency of the recording instrument to handle the specified gain ranges used in the field which leads to clipping of the data sets.

Variables used in VEL_ANAL :

Cvs(959) : Real format array - the constant velocity corrected data is added to this array to produce the final stacked trace(s).

Final_cvs (959) : Integer format array - Prior to writing the CVS data to disc it is transformed to integer format to save disc space.

Cvg(959) : Integer format array - Every constant velocity NMO corrected trace is sotred in this array. If the CVG option was selected this array is written to disc after NMO correction.

T(959) : Real format array - Calculated NMO values are stored in this array.

Anal_chce$: Either "CVG" or "CVS" - selected output option.

Si : Sample interval in seconds (as opposed to Sample_interval which is in msec).

Nr_midpoints : The number of midpoints that must be stacked to form the CVS display.

Zero_trace_ques$: Either "Y" or "N" - determines whether the zero offset trace must be included in the stack or not.

Count : The number of previously calculated NMO files on disc.

Prev_nmo$(*) : Alphanumeric format array - The corresponding filenames of the previously calculated NMO files.

Seperation : The trace offset (source - geophone seperation)

T1,T2 : The two way times (multiples of the sample interval) which straddles the calculated NMO value. (in array T(*))

Index1,Index2 : The corresponding sample numbers of T1 and T2.

Remainder : 0 < Remainder < sample interval - The time (in msec) that the calculated normal moveout (in array T(*)) is greater than T1. It is used to do the linear interpolation between T1 and T2 if necessary.

Difference : The difference between T1 and the interpolated sample value.

II.3.5 NMO_CORR_1

This program applies NMO corrections to CMP gathers prior to stacking the gather. The algorithm and variables used in NMO_CORR_1 is the same as those used in VEL_ANAL. The main difference is that a velocity function as opposed to a constant velocity is used to correct the gather. The velocity function is picked using VEL_ANAL. Program

NMO_CORR_1 prompts for two-way time - velocity pairs and interpolates a velocity value for every sample point. It is important to note that the first and last two-way time - velocity pairs entered must be at 0 and 192 msec (the sweep length) respectively. Because velocity functions will differ from processing run to processing run no calculated normal moveout values are stored on disc. The program has its own disc reading routine and static corrections are applied prior to NMO correction and stack. If so required by the user the pre-stack traces can be stored on disc as well.

As soon as the NMO corrections have been applied the program displays the following plotting options :

i. Plot NMO corrected traces.

ii. Plot NMO functions.

Option 1 - Plot NMO corrected traces : This routine displays the NMO corrected traces and prompts the user to specify a mute zone to mute out the first breaks and NMO stretch.

Option 2 - Plot NMO function : This routine displays the NMO functions used to correct the CMP gather for normal moveout. Only the input two way time - velocity pairs are displayed and not the interpolated values of the velocity function.

Variables used in NMO_CORR_1 :

NMO(24,959) : Same as T(*) in VEL_ANAL with the only difference being sized for 24 traces.

Mute(24) : Integer format array - the number of samples corresponding to the calculated mutes on every trace.

Store_trace(959) : Integer format array - NMO corrected traces (unstacked) that will be written to disc if so required by the user.

Cmp_store(959) : Integer format array - stacked trace that is written to disc.

Twt(8), Input_vel(8) : Real format arrays - the velocity function base points are entered into these arrays.

Stack_vel(959) : The interpolated velocity function. Every element is the velocity value that will be used to correct the corresponding sample number for normal moveout.

Nmo_corr_trace(24,959) : Real format array - the pre-stack NMO corrected traces.

Cmp_stacked(959) : Real format array - the final stacked trace.

Diff : Used for interpolation in two different contexts. The first time in the velocity interpolation routine where the difference between two consecutive sample

numbers' velocities are assigned to "diff" and the second time in the interpolation of the NMO corrected sample value.

Distance : the source-geophone offset of the trace being muted.

Max_offset : the source-geophone offset of the outer trace of the CMP gather.

Begin : Either 1 or 2 - the first trace that must be included in the stack.

II.3.6 NMO_CORR

This program is identical to NMO_CORR_1 with two exceptions :

i. There are no plotting routines (which implies a prior knowledge of the mute zones are needed).

ii. The error trap routine has a routine which switches between mass storage units when "disc full" errors are encountered.

The suggested approach to NMO correcting and CMP stacking is to do a few corrections using NMO_CORR_1 to determine the mute zones and then use NMO_CORR to mute and stack the rest of the line.

II.3.7 FILTR_BUTR

This recursive Butterworth filter algorithm was adapted from Fatti's PhD thesis (1974). The transfer function of the filter is defined as a Z-transform and is the product of a number of polynomial fractions A(z), B(z), C(z) .... where A(z), B(z), etc. of the form

$$A(z) = a_0 + a_1 z + a_2 z^2 + a_3 z^3 + ...a_n z^n$$

is.

The constants a, b, c, d, e depend on the sampling interval, high and low cut frequencies, the order of the filter and the polynomial fraction being calculated.

The order of the filter is a measure of the number of polynomial fractions and determines the steepness of the cut-off slopes. The higher the order of the filter the steeper the cutoff slopes. To avoid possible instability caused by roundoff errors a cascaded filter is used. The input is filtered recursively with polynomial fraction A(z) and then this output with B(z) and so forth rather than using the multiplied fractions A(z)B(z)C(z) to produce one large fraction which can lead to round-off errors.

Recursive filters are physically realizable in that they cannot respond to any input before it occurs. Per definition then they cannot be zero phase. However, zero-phase filtering is achieved as discussed in the paragraph on recursive filtering. To allow the first output to decay to a reasonably small amplitude before terminating the computation a sufficient number of zeros must be added to the input

data.

Variables used in FILTR_BUTR :

Coeff(5,7) : Real format array - The calculated Butterworth coefficients of the Z-transform

Fins(1163), Outs(1163) : Real format array - Originally the field data is written into array Fins(*), filtered recursively and written into array Outs(*). The array Outs(*) is reversed in time filtered again with the same filter to convert to zero phase.

Fin(1163) : Real format array - The final filtered data is written into this array.

Int_trace(959) : Integer format array - The filtered data is written into integer format before writing to disc.

F1, F2 : The lower and upper frequencies of the filter passband (corresponding to 6 dB down points (50% amplitude).

Order : The order of the Butterworth filter

Delt : The sample interval in seconds

II.4 INTERPRETATION PROGRAMS

II.4.1 RAY_TRACE

This program traces a seismic ray from a specified source position to a number of specified receiver positions. A geological model is inputted by the user and using Snell's law at the interfaces refraction angles are calculated. The user also specifies the accuracy at the geophone positions required. The more accurate the "target zone" specified at the geophone positions the longer the run time of the program. If the ray exit position pass over the last specified geophone position the increment angle is reduced and the whole ray tracing algorithm repeated.

Variables used in RAY_TRACE :

Total_time(25,10) : Real format array - the calculated arrival times for the different interfaces (max = 10) for every geophone (max = 25).

Geo(25) : Real format array - The ground positions (offset from source) of the geophones.

Screen(7500) : Integer format graphics array - The model plus drawn rays are stored in this array.

Accuracy : The accuracy required at the geophone positions.

Incident_d(10), Incident_u(10) : Downgoing and returning (up) wavefronts' incident angles.

Transmitted_d(10), Transmitted_u(10) : Downgoing and returning wavefronts' transmitted angles

II.4.2 SYNTH_CMP

This program generates synthetic common midpoint gathers and was developed to illustrate velocity analysis and stacking routines. The program prompts for a random seed number which will ensure that the random noise added to the traces are different - the random number generating function on the Hewlett Packard initializes to the same value every time a program is run. A rapid decaying wavelet is generated and convolved with a spiked CMP gather if so desired. The normal moveout of the specified events are calculated using the normal moveout equation mentioned in the normal moveout paragraphs in the main text.

Variables used in SYNTH_CMP :

The variables used in this program are self explanatory.

II.4.3 AMP_SPECTR

This program calculates the amplitude spectrum of a specified trace in the common memory block. The user is prompted for the trace number of the input trace. The trace is then displayed and an FFT performed on

the trace. The calculated amplitude spectrum is normalized to 1 and displayed on screen. The user can then either dump the graphics to the printer, calculate a new amplitude spectrum of one of the other traces in memory or return to the main menu. This program was adapted from a frequency-domain filtering routine of E.H. Stettler to test and evaluate the filtering algorithm used in FILTR_BUTR. By switching between FILTR_BUTR, AMP_SPECTR and DISP_SEIS the best band-pass filter can be selected to 'clean up' the data.

Variables used in AMP_SPECTR :

The variables in AMP_SPECTR are either self explanatory or used only for calculation purposes.

APPENDIX III

EXAMPLES

EXAMPLES

III.1 GENERAL DISCUSSION

Both real and synthetic data are used to demonstrate the different routines that were developed. Six seismic lines were shot near SASOL's Sigma mine at Sasolburg and two lines near Secunda. Different parts of different lines were selected to illustrate the various routines to its best. Synthetic data were generated and are used to demonstrate the velocity analysis and stacking routines as no field data with enough identifiable reflectors could be found to do proper velocity analysis and to illustrate the effectiveness of this method to its best. The importance of correct static corrections can also be seen on these examples.

III.2 AMPLITUDE RESTORATION AND MANIPULATION EXAMPLES

A constant offset section of 23 shots on seismic line B near Secunda is used to illustrate the amplitude manipulation routines. A forty five meter constant offset section was read using READ_SEIS. The unprocessed section is shown in figure 7. All the amplitude manipulation was done using program AMPLITUDE.

As no amplitude clipping is observed on the first breaks, no mute was applied to the data. Time TO (reference time at which all amplitude processe will start) was taken as 40 msec for all the traces. This corresponds more or less to the time of the first energy arrival on all the traces. The exponential option was used to correct for spherical divergence. After a few trials in which parameters 'K' and 'a' were varied (see paragraph on factors that influence amplitude in

text) values of K=8 and a=0.5 were selected as the parameters that balances the traces the best. Note that the K-value is nothing more than a constant gain factor applied to the trace to compensate for the overall reduction of amplitude that the trace undergoes. This overall reduction of amplitude is understood if one remembers that every sample is multiplied by it's corresponding two-way time (in seconds) as well as the exponential factor (if the exponential spherical divergence correction is used). As the total sweep length of the data is 0.192 s, every amplitude is multiplied by an increasing fraction which, although reducing the overall amplitude, boosts the later amplitudes.

Looking at figure 8, the equalization within a trace can be seen after the spherical divergence correction has been applied. The trace that bests illustrate the effectiveness of the spherical divergence correction is trace 2 (shot 250).

To correct the very noticable differences in amplitudes between traces (due to different shooting or field techniques), the traces were normalized with respect to a reference amplitude which is specified in the program. Figure 9 demonstrates the effectiveness of this trace normalization quite strikingly.

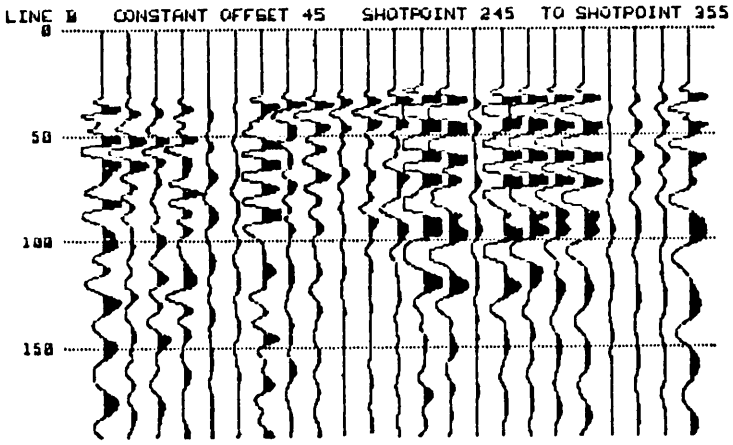


Figure 7 Unprocessed field data

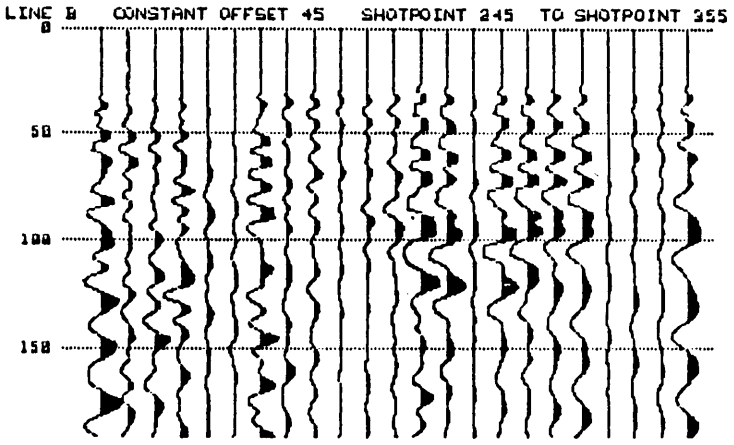


Figure 8 Spherical divergence corrected

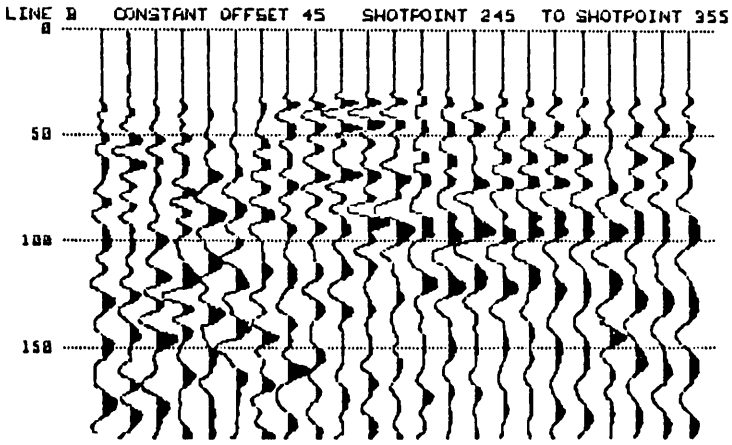


Figure 9 Normalized

III.3 STATIC EXAMPLES

Correcting for the delays in traveltime due to inhomogeneous nature of the near surface is one of the most important processes in high resolution seismic work. The unprocessed field data is shown in figure 10 and the amplitude corrected traces in figure 11. After the static correction have been applied the horizontal line up of events are much easier as can be seen in figure 12.. This static corrected data was filtered with an open band-pass filter of 10 to 110 Hz to see what the corrected traces look like if some of the higher frequency components have been removed. The filtered traces in figure 13 show better continuity of the event at +/- 40 msec in the vicinity of trace eight. Whether the unfiltered data is closer to the truth or not - the discontinuity caused by some minor disturbance or incorrect static correction is difficult to say.

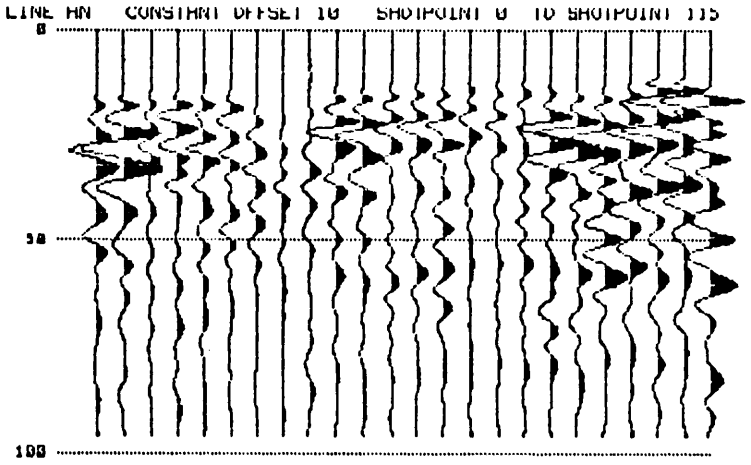


Figure 10 Unprocessed field data

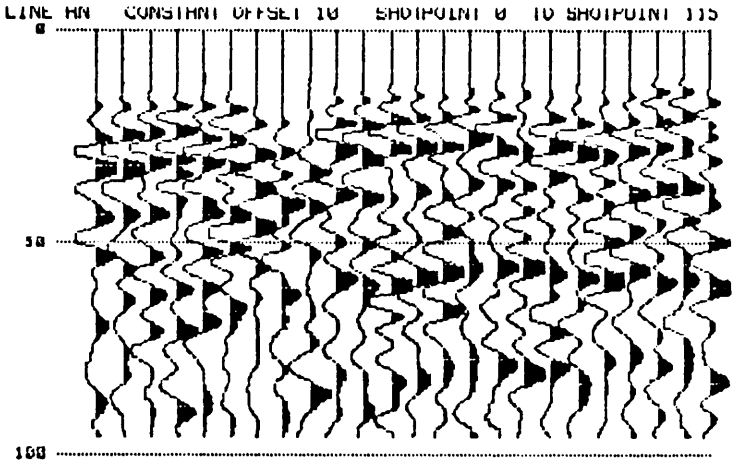


Figure 11 Amplitude corrected

EXAMPLES

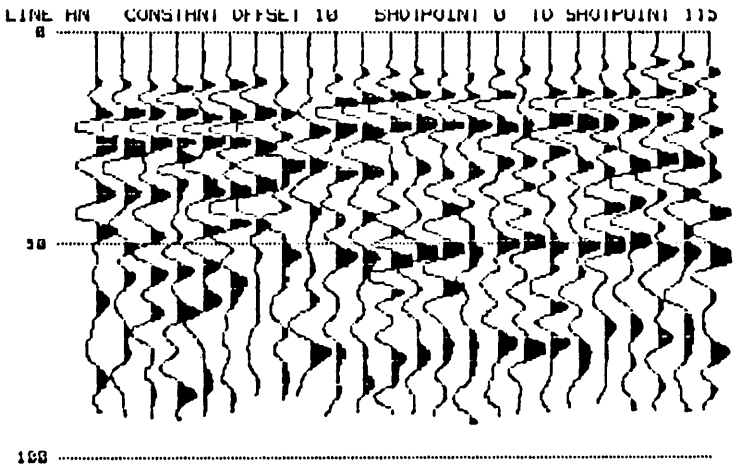


Figure 12 Static correction applied

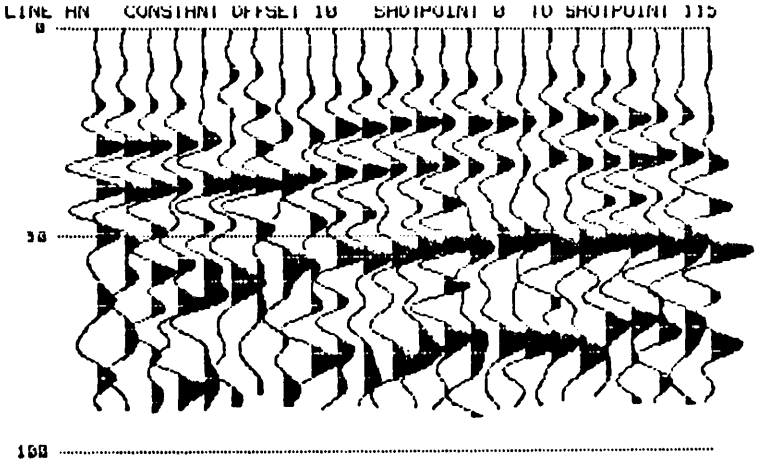


Figure 13 Filtered data

III.4 VELOCITY ANALYSIS AND NMO CORRECTION EXAMPLES

Five synthetic CMP gathers, generated by SYNTH_CMP are used to demonstrate various aspects of aspects of velocity analysis and NMO-correction using programs VEL_ANAL,NMO_CORR_1 and NMO_CORR.

The five synthetic CMP's that were generated, each with six reflectors differing only slightly in two-way time and RMS-velocity, have different random noise added to each trace. The five CMP gathers are shown in figure 14.

Synthetic CMP QS_55 (12 fold gather) is used to demonstrate the CVG method of velocity analysis, while CMP's QS_55 to QS_75 (8 fold) were used to demonstrate the CVS method of velocity analysis. The velocities used range from 1200 m/s to 2400 m/s in 200 m/s increments. The CVS panels are displayed next to the corresponding CVG panels in figure 15 on pages III-11 to III-13.

A velocity function was picked from these panels using the following reasoning : The event at 15 msec is overcorrected in the 1200 m/s panels. On the 1400 m/s panels this event lines up on the CVG and the stacked wavelet is the sharpest on the corresponding CVS panel. A velocity of 1400 m/s were therefore picked for the event at 15 msec. The event at 35 msec is still slightly overcorrected on the 1600 m/s panel, but under-corrected on the 1800 m/s panel. A velocity of 1650 m/s was interpolated as the velocity that will give the best results in the stack. Using the same reasoning the velocities for the other events were picked. The picked velocity function and associated NMO

functions are displayed in figure 16. The NMO corrected CMP gather QS_55 is displayed in figure 17.

An expanded display of the upper part of the NMO corrected CMP gather is given in figure 19. After applying a stretch mute to the gather (shown in figure 20) the CMP gather was stacked and compensated for loss of fold in the upper part of the trace as discussed in the velocity analysis paragraph (figure 18). The five CMP's were then stacked using the same velocity function and stretch mutes selected for QS_55. The stacked section is shown in figure 21.

The CVG and CVS panels were generated by VEL_ANAL and displayed with READ_SEIS and DISP_SEIS. The velocity function, NMO functions and corrected CMP gather before and after mute were generated by NMO_CORR_1 and dumped to the printer. The stacked CMP's QS_55 to QS_75 were generated by NMO_CORR and displayed using READ_SEIS and DISP_SEIS.

EXAMPLES

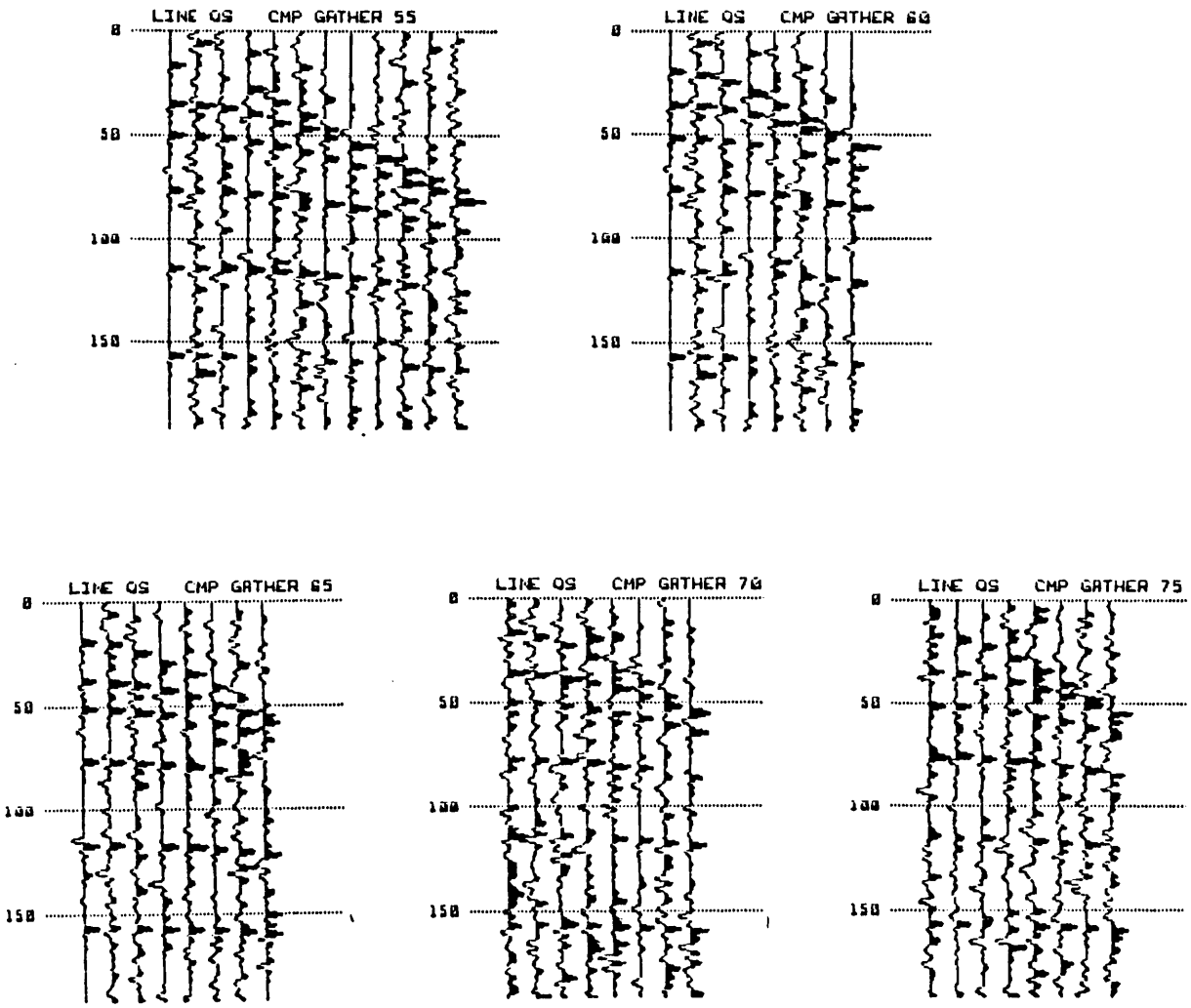


Figure 14 Synthetic CMP gathers

EXAMPLES

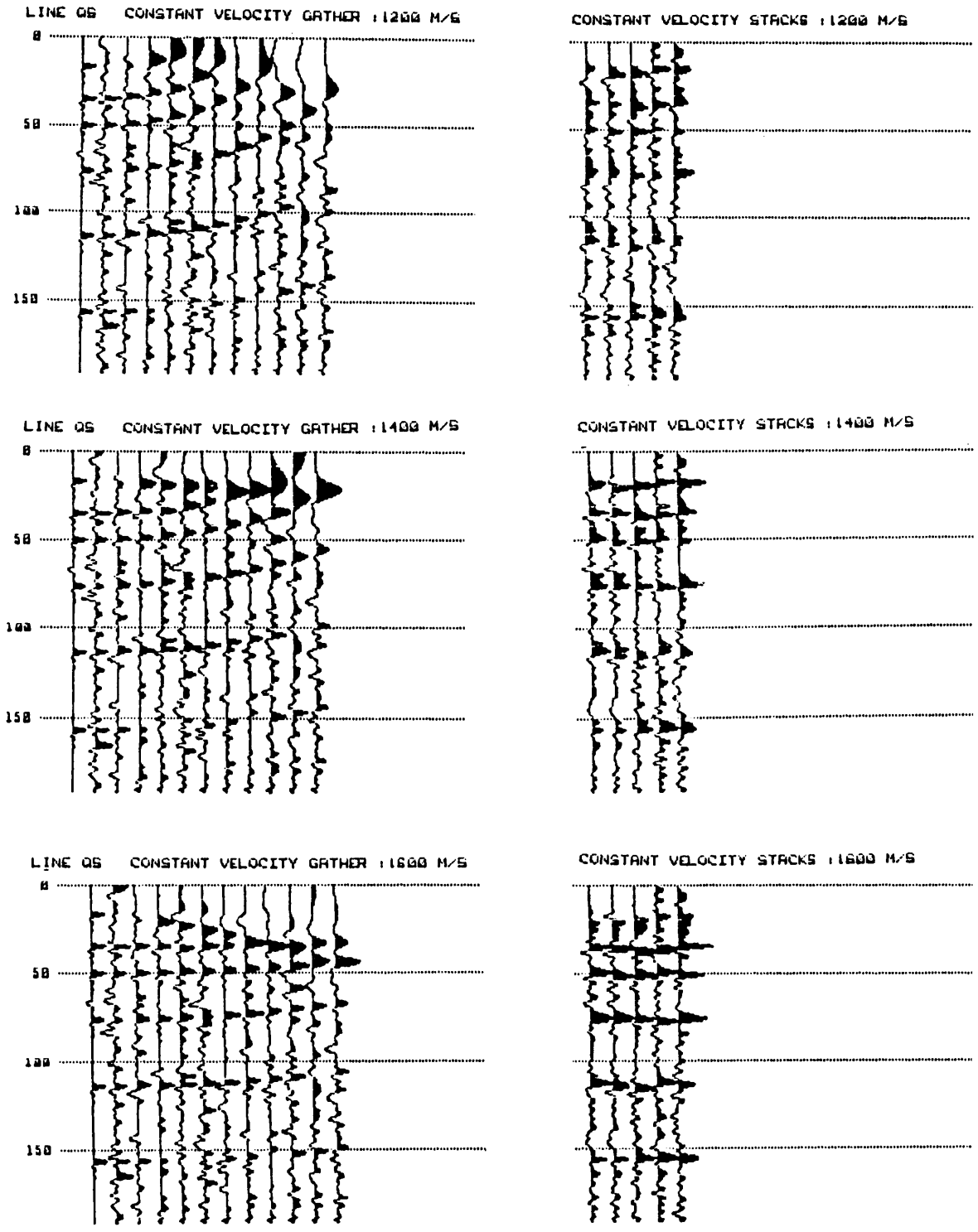
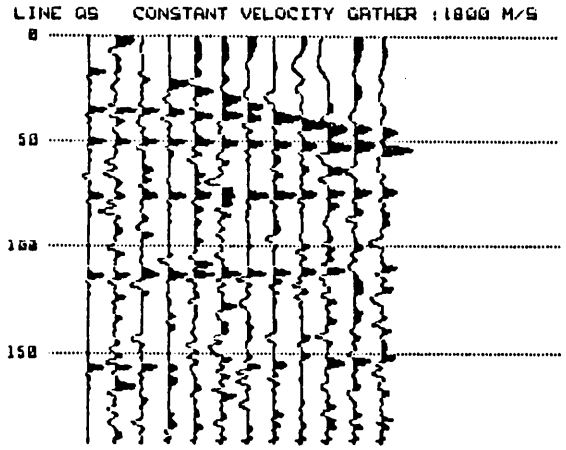
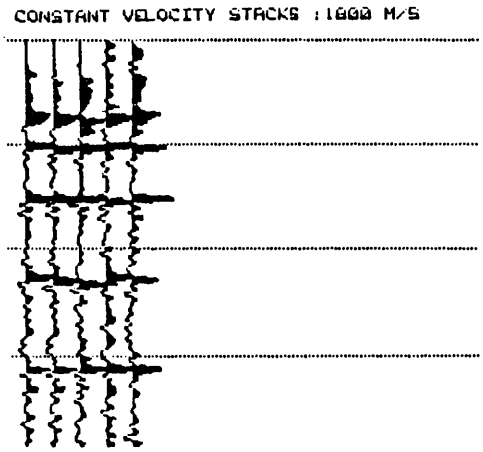
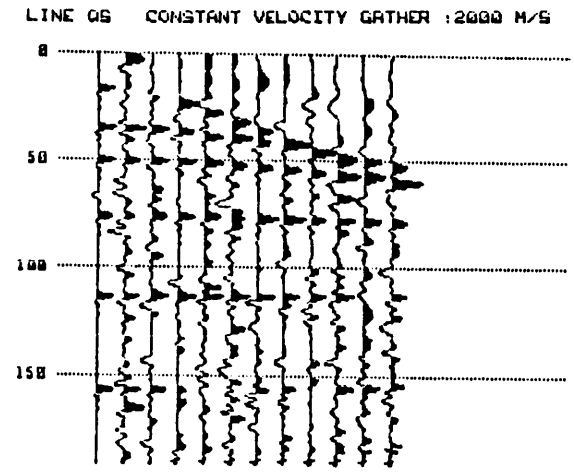


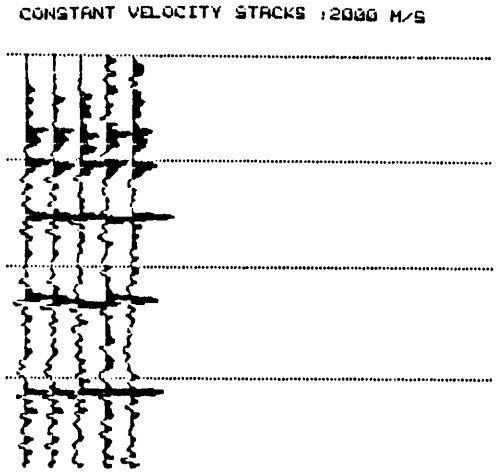
Figure 15 Velocity analysis panels

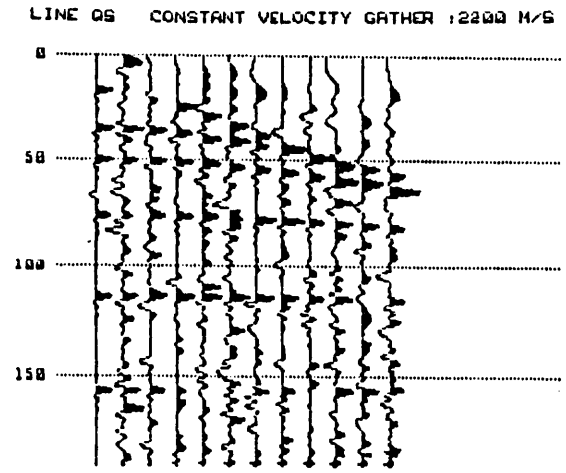
EXAMPLES

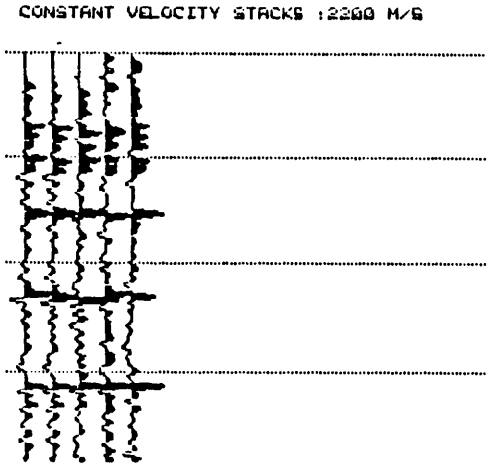




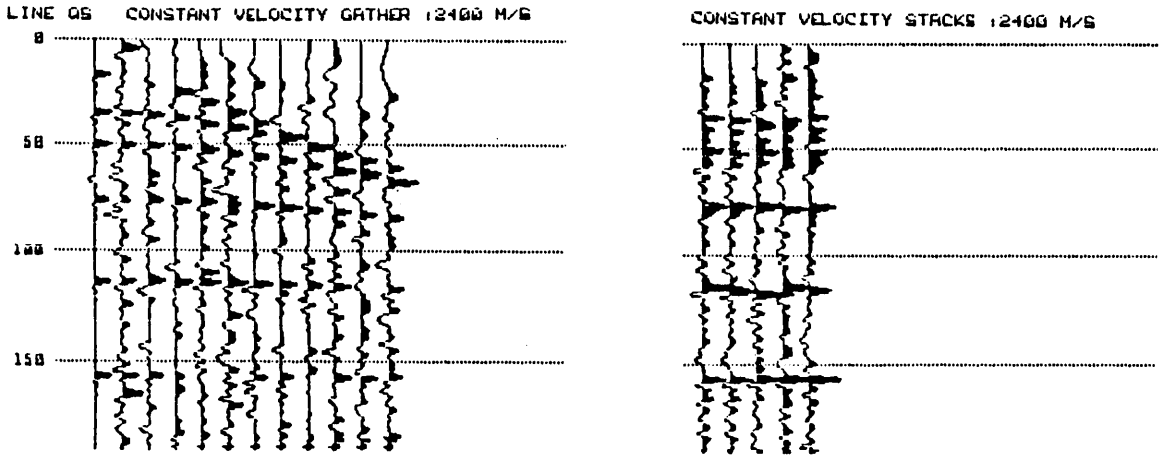








EXAMPLES



EXAMPLES

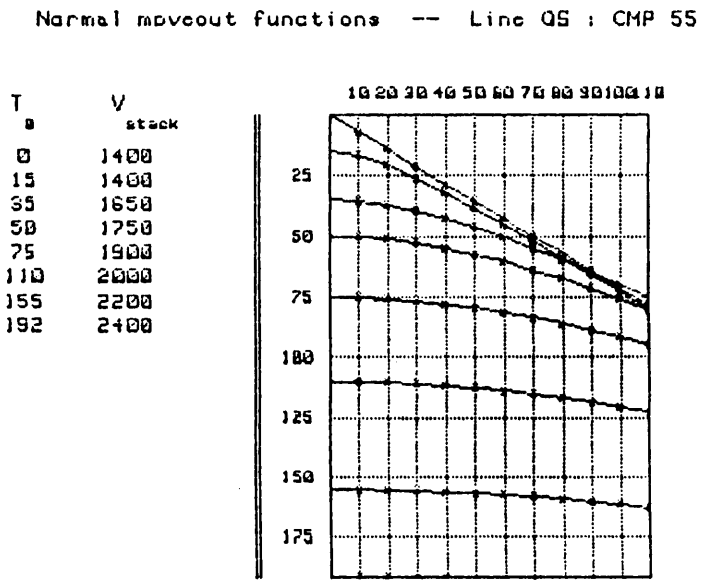


Figure 16 Picked Velocity function

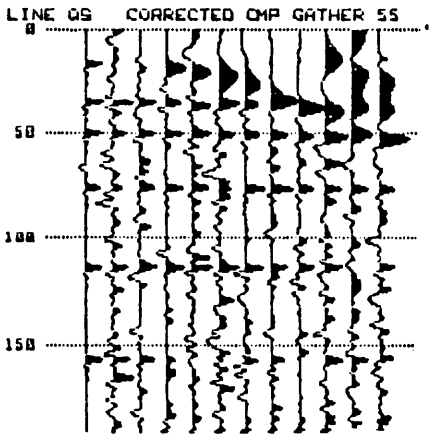


Figure 17 NMO corrected gather

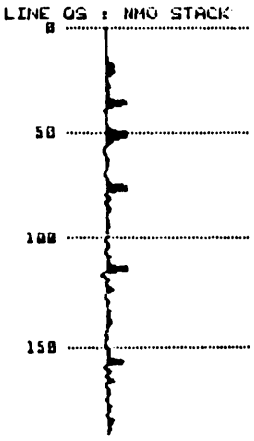


Figure 18 Stacked CMP

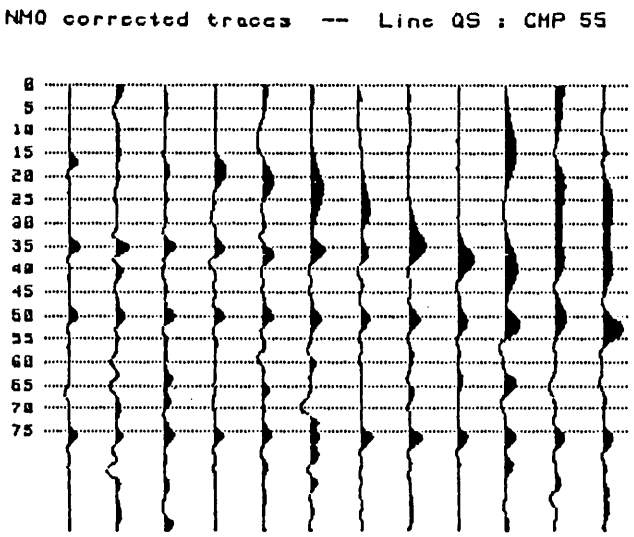


Figure 19 Expanded view – NMO corrected gather

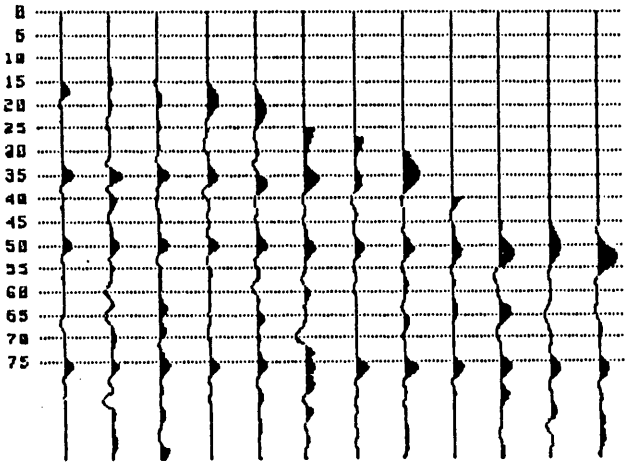


Figure 20 Stretch mute applied

EXAMPLES

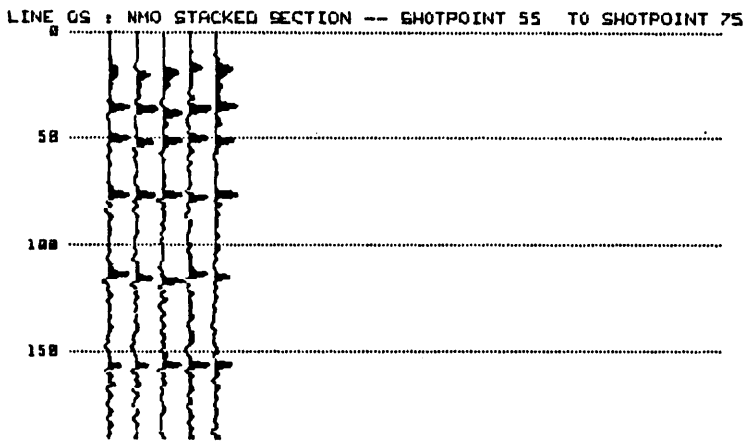


Figure 21 Stacked synthetic gathers

III.5 FILTERING AND SCALING EXAMPLES

Constant offset traces from line AS near SASOL's Sigma coal mine are used to illustrate the effect of filtering and scaling on the data set. The example also illustrate the importance of high frequency content in the signal in high resolution seismics. The input field data is shown in figure 22 with the amplitude spectrum of the second trace (shot 5) in figure 23. This set of 24 traces were corrected for loss of amplitude and normalized with respect to one another. The normalized data is shown in figure 24. The first eight traces show wide frequency bands and were selected to illustrate the effect of filtering on the data. The wiggle traces next to the amplitude spectrums illustrates the effect of the filter on the wavelet shape. All the data was filtered with the recursive Butterworth filter and normalized after filtering using program FILTR_BUTR. The plots were done using DISP_SEIS. The amplitude spectrum of the second trace in the display was generated and displayed with AMP_SPECTR.

The first filter used was a fourth order Butterworth filter with a very open band-pass of 5 Hz to 100 Hz. The resultant data set is shown in figures 25. The events at 50 msec is a lot clearer and easier to follow across the traces in these examples. The high frequency components have allready been severly attenuated. The next set of examples (figure 26) is a eighth order filter using the same cutoff frequencies. The biggest difference is noted just above 50 msec where the higher frequency components have been attenuated to a higher degree. As mentioned in the paragraph dealing with filtering,

higher order recursive filters lead to sharper cutoff slopes which is clearly illustrated if one looks at the corresponding amplitude spectrum.

Figure 27 are the result of filtering the same data set with a fourth order 8 to 80 Hz filter and figure 28 filtering with a fourth order 15 to 75 Hz filter. The importance of high frequency content is clearly illustrated in these examples.

Comparing the positive black loop of the event at 50 msec on the different data sets shows that the phase of the data is unaffected by the filtering operation. The filter thus approximates a zero phase filter as described in the paragraph on filtering.

EXAMPLES

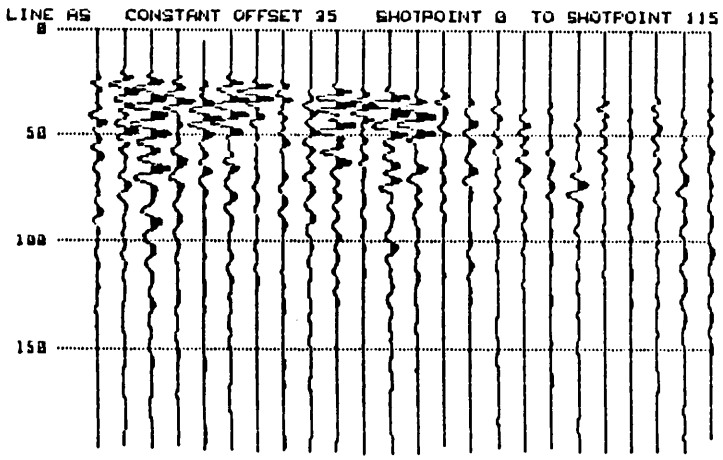


Figure 22 Unprocessed field data

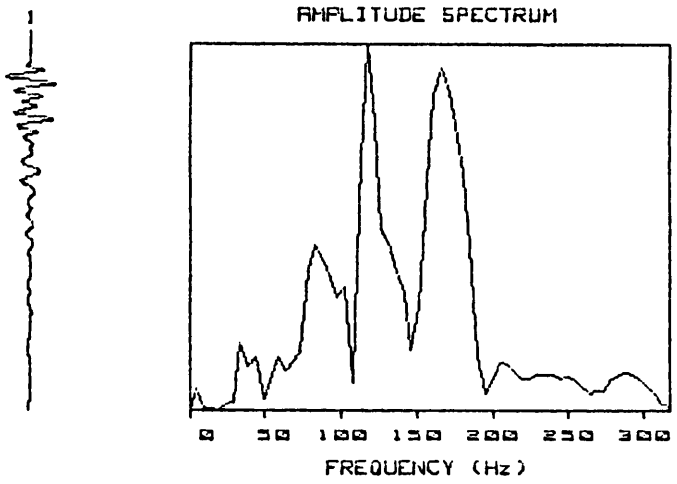


Figure 23 Amplitude spectrum of second trace

EXAMPLES

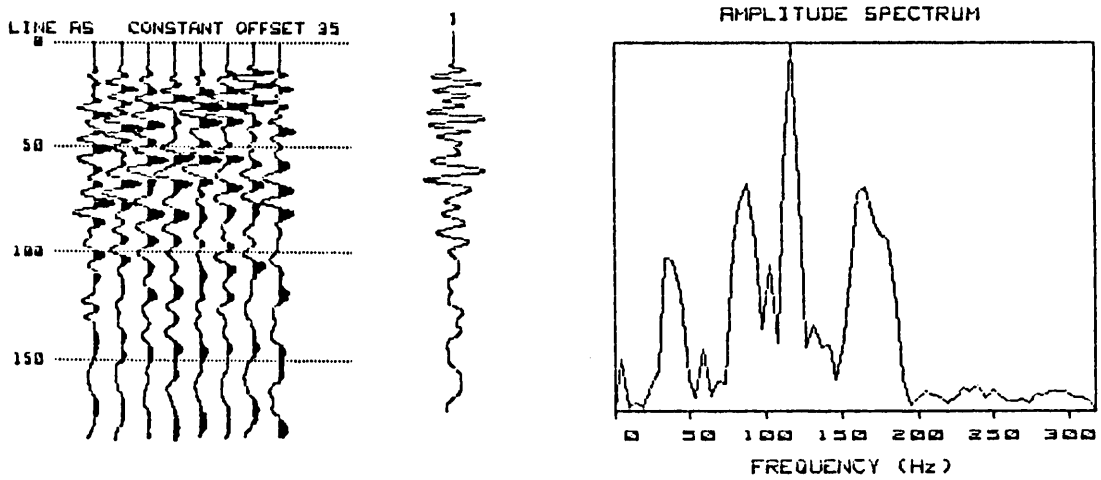


Figure 24 Normalized data

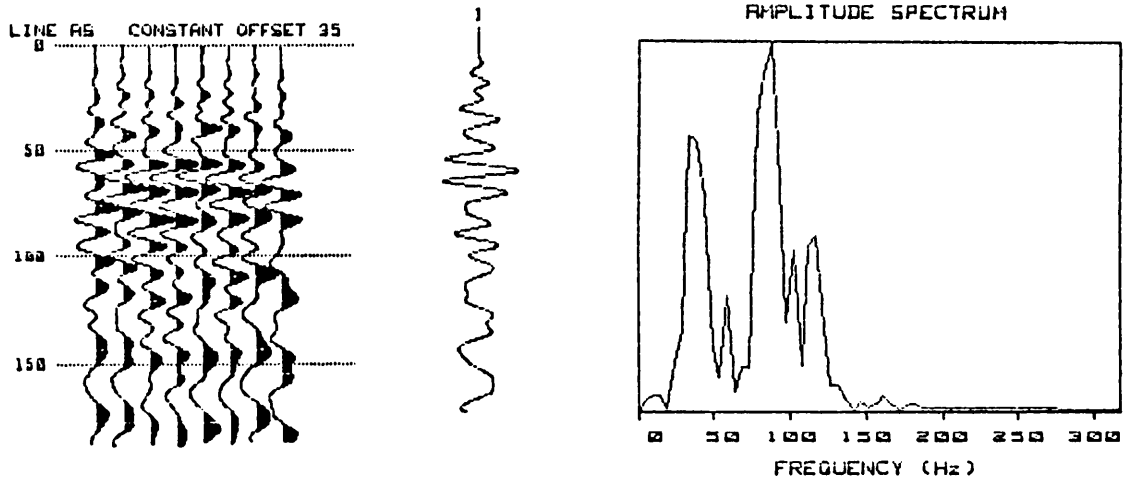


Figure 25 4th order Butterworth filter 5 – 100 Hz

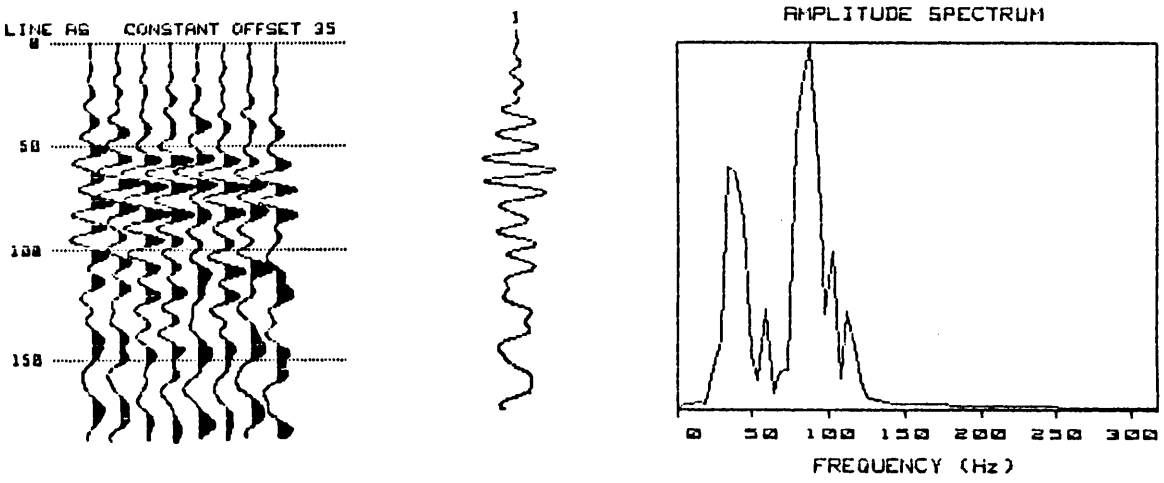


Figure 26 8th order Butterworth filter 5 – 100 Hz

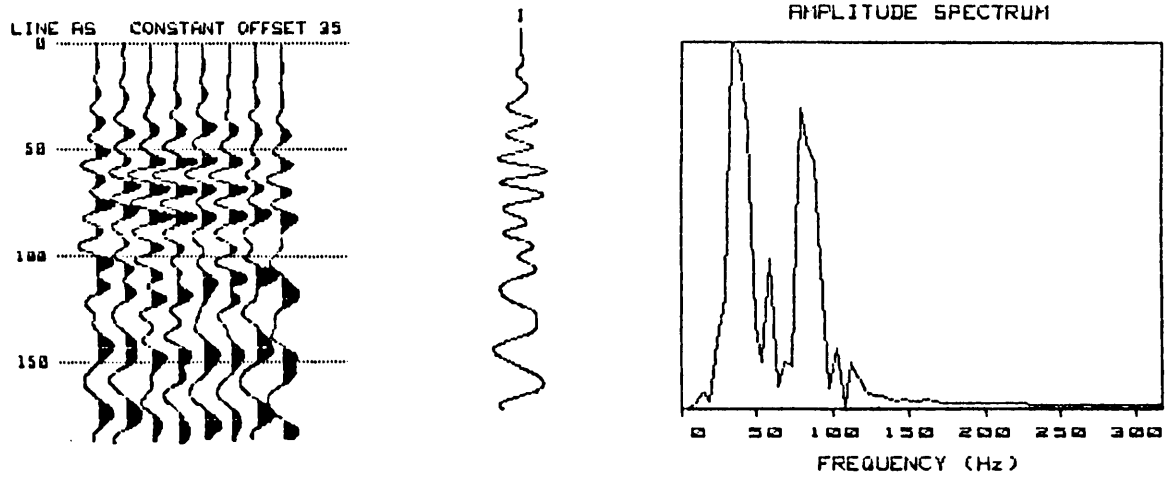


Figure 27 4th order Butterworth filter 8 – 80 Hz

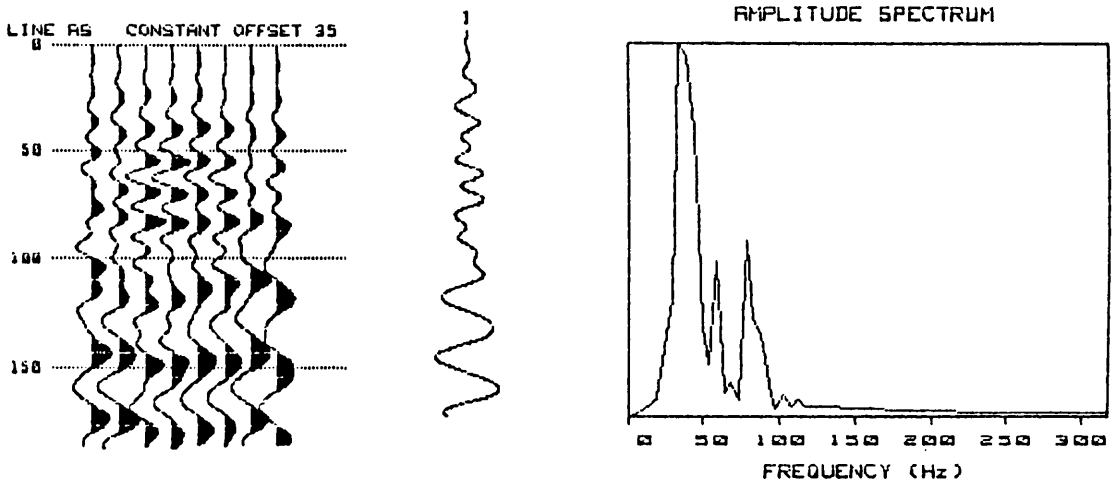
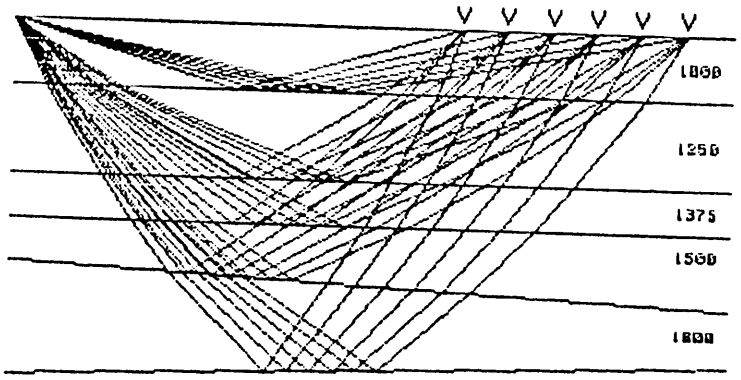


Figure 28 4th order Butterworth filter 15 - 75 Hz

III.6 RAY TRACING EXAMPLE

Using program RAY_TRACE ray tracing through a simple geological model was done. The resultant graphic display is shown in figure 29. This routine is quite time intensive.

EXAMPLES



SEISMIC MODELING :

Model :

Depth	Dip	Velocity
15 m	0	1000 m/s
35 m	0	1250 m/s
45 m	0	1375 m/s
55 m	-5	1500 m/s
80 m	4	1800 m/s

REFLECTION TIMES :

Offset	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5
50	.0582	.0760	.0873	.1008	.1216
55	.0627	.0785	.0895	.1027	.1226
60	.0670	.0813	.0916	.1047	.1237
65	.0715	.0843	.0940	.1069	.1249
70	.0762	.0873	.0965	.1090	.1262
75	.0808	.0902	.0990	.1114	.1275

Figure 29 Ray tracing example

APPENDIX IV

PROGRAM LISTINGS

```
10    !  AUTOST  --  Main seismic menu program
20    !
30    !  November 1986   ---   U.P.  ---  A.T.Dippenaar
40    !
50    !  Last changes on 18 January 1988
60    !=============================================================================
70    !
80    OPTION BASE 1
90    !
100   COM /Seis_int/ Line$[2],INTEGER Trace(24,960),Opt,Amt,Gain(24)
110   COM /Seis_real/ REAL Shot(24),Geo(24),Shift_value(24),Offset
120   COM /Processed/ REAL Processed_trace(24,959),Pointy,V
130   COM /Polar/ INTEGER Pol_factor(24)
140   COM /Statics/ REAL Geophone_static(100),Source_static(100),Peg(100)
150   COM /Screens/ INTEGER Screen_1(7500),Screen_2(7500),REAL X,Y,Gain_amplitud
e,Gain_time
160   !
170   !=============================================================================
180   DIM Marker$[6]
190   Marker$="==>"&CHR$(8)&CHR$(8)&CHR$(8)
200   IF Pointy=0 THEN Pointy=7
210   Chce=0
220   MASS STORAGE IS ":,700,0"
230   CONTROL 1,12;1
240   GCLEAR
250   OUTPUT KBD;"K";
260   !
270   GOSUB Main
280   ON KBD GOSUB Keyboard
290   ON KNOB .15 GOSUB Move_echo
300 Spin:GOTO Spin
310   !*****************************************************************************
320 Main:  !
330   Max_pointy=17
340   OUTPUT KBD;"K";
350   PRINT TABXY(34,2);CHR$(129)&" Main menu "&CHR$(128)
360   PRINT TABXY(22,5);" 1 ..... Sort traces from mass-stored disc "
370   PRINT TABXY(22,7);" 2 ..... Read field traces from disc "
380   PRINT TABXY(22,9);" 3 ..... Read processed traces from disc "
390   PRINT TABXY(22,11);" 4 ..... Processing routines  "
400   PRINT TABXY(22,13);" 5 ..... Display traces     "
410   PRINT TABXY(22,15);" 6 ..... Interpretation routines"
420   PRINT TABXY(22,17);" 7 ..... EXIT program"
430   PRINT TABXY(19,Pointy);Marker$;
440   RETURN
450   !-----------------------------------------------------------------------------
460 Move_echo:  !
470   PRINT TABXY(19,Pointy);"   ";
480   IF KNOBX>0 THEN
490     Pointy=Pointy+2
500     IF Pointy>Max_pointy THEN Pointy=Max_pointy
510   ELSE
520     Pointy=Pointy-2
530     IF Pointy=3 THEN Pointy=5
540   END IF
550   PRINT TABXY(19,Pointy);Marker$;
560   RETURN
570   !-----------------------------------------------------------------------------
580 Keyboard:   !
590   SELECT KBD$
```

```
600   CASE CHR$(255)&"^"
610     PRINT TABXY(19,Pointy);"    ";
620     Pointy=Pointy-2
630     IF Pointy=3 THEN Pointy=Max_pointy
640     PRINT TABXY(19,Pointy);Marker$;
650   CASE CHR$(255)&"V"
660     PRINT TABXY(19,Pointy);"    ";
670     Pointy=Pointy+2
680     IF Pointy>Max_pointy THEN Pointy=5
690     PRINT TABXY(19,Pointy);Marker$;
700   CASE CHR$(255)&"E",CHR$(255)&"X",CHR$(255)&"C"
710     SELECT Chce
720   !=============================================================================
730   !  Main menu
740   !=============================================================================
750     CASE 0
760       SELECT Pointy
770       CASE 5
780         LOAD "BISON_SORT"
790       CASE 7
800         Chce=1
810         Pointy=5
820         GOSUB Read_field
830       CASE 9
840         Chce=2
850         Pointy=5
860         GOSUB Read_processed
870       CASE 11
880         Chce=3
890         Pointy=5
900         GOSUB Process_menu
910       CASE 13
920         Chce=4
930         Pointy=5
940         GOSUB Display_menu
950       CASE 15
960         Chce=5
970         Pointy=5
980         GOSUB Interpret_menu
990       CASE 17
1000        GCLEAR
1010        OUTPUT KBD;"K";
1020        CONTROL 1,12;0
1030        STOP
1040      END SELECT
1050  !=============================================================================
1060  !  Unprocessed field traces reading options
1070  !=============================================================================
1080    CASE 1
1090      SELECT Pointy
1100      CASE 5
1110        Opt=1
1120        LOAD "READ_SEIS"
1130      CASE 7
1140        Opt=2
1150        LOAD "READ_SEIS"
1160      CASE 9
1170        Opt=3
1180        LOAD "READ_SEIS"
1190      CASE 11
```

```
1200        Chce=0
1210        Pointy=7
1220        GOSUB Main
1230      END SELECT
1240 !=============================================================================
1250 !  Processed traces reading options
1260 !=============================================================================
1270    CASE 2
1280      SELECT Pointy
1290      CASE 5
1300        Opt=4
1310        LOAD "READ_SEIS"
1320      CASE 7
1330        Opt=5
1340        LOAD "READ_SEIS"
1350      CASE 9
1360        Opt=6
1370        LOAD "READ_SEIS"
1380      CASE 11   '
1390        Opt=7
1400        LOAD "READ_SEIS"
1410      CASE 13
1420        Pointy=9
1430        Chce=0
1440        GOSUB Main
1450      END SELECT
1460 !=============================================================================
1470 !  Processing options
1480 !=============================================================================
1490    CASE 3
1500      SELECT Pointy
1510      CASE 5
1520        LOAD "POLAR"
1530      CASE 7
1540        LOAD "AMPLITUDE"
1550      CASE 9
1560        LOAD "STATICS"
1570      CASE 11
1580        LOAD "VEL_ANAL"
1590      CASE 13
1600        OUTPUT KBD;"K";
1610        PRINT TABXY(5,14);"        Do you want to stack more than one CMP [
DEF = Y ]",Stack_ques$
1620        PRINT TABXY(5,16);"If you want to view the corrected traces to speci
fy the mute answer N"
1630        PRINT TABXY(5,18);"     Answering Y assumes you have a knowledge of
the mute zone"
1640        Stack_ques$="Y"
1650        INPUT Stack_ques$
1660        IF Stack_ques$="Y" THEN
1670          LOAD "NMO_CORR"
1680        ELSE
1690          LOAD "NMO_CORR_1"
1700        END IF
1710      CASE 15
1720        LOAD "FILTR_BUTR"
1730      CASE 17
1740        Chce=0
1750        Pointy=11
1760        GOSUB Main
```

```
1770      END SELECT
1780 !=============================================================================
1790 !  Displaying options
1800 !=============================================================================
1810    CASE 4
1820      OUTPUT KBD;"K";
1830      SELECT Pointy
1840      CASE 5
1850        LOAD "DISP_SEIS"
1860      CASE 7
1870        GOSUB Screen_test
1880        IF Screen_test=1 THEN
1890          DUMP DEVICE IS 701
1900          GRAPHICS ON
1910          GLOAD Screen_2(*)
1920          DUMP GRAPHICS
1930        END IF
1940        GRAPHICS OFF
1950        GOTO 2100
1960      CASE 9
1970        GOSUB Screen_test
1980        IF Screen_test=1 THEN
1990          DUMP DEVICE IS 701,EXPANDED
2000          GRAPHICS ON
2010          GLOAD Screen_2(*)
2020          DUMP GRAPHICS
2030          DUMP DEVICE IS 701
2040        END IF
2050        GRAPHICS OFF
2060        GOTO 2100
2070      CASE 11
2080        LOAD "PLOT_SEIS"
2090      CASE 13
2100        Chce=0
2110        Pointy=13
2120        GOSUB Main
2130      END SELECT
2140 !=============================================================================
2150 !   Interpretation options
2160 !=============================================================================
2170    CASE 5
2180      SELECT Pointy
2190      CASE 5
2200        LOAD "SYNTH_CMP"
2210      CASE 7
2220        LOAD "RAY_TRACE"
2230      CASE 9
2240        LOAD "AMP_SPECTR"
2250      CASE 11
2260        Chce=0
2270        Pointy=15
2280        GOSUB Main
2290      END SELECT
2300    END SELECT
2310  CASE ELSE
2320    BEEP 2200,.15
2330  END SELECT
2340  RETURN
2350  !*****************************************************************************
2360 Read_field:!
```

```
2370  OUTPUT KBD;"K";
2380  PRINT TABXY(28,2);CHR$(129)&" Field trace read menu "&CHR$(128)
2390  PRINT TABXY(22,5);" 1 ..... Get a field record"
2400  PRINT TABXY(22,7);" 2 ..... Gather uncorrected CDP traces
2410  PRINT TABXY(22,9);" 3 ..... Gather a common offset section"
2420  PRINT TABXY(22,11);" 4 ..... RETURN to main menu"
2430  PRINT TABXY(19,Pointy);Marker$;
2440  Max_pointy=11
2450  RETURN
2460  !-------------------------------------------------------------------------
2470 Read_processed: !
2480  OUTPUT KBD;"K";
2490  PRINT TABXY(26,2);CHR$(129)&" Processed trace read menu "&CHR$(128)
2500  PRINT TABXY(22,5);" 1 .... Constant velocity NMO corrected traces (CVG pan
els)"
2510  PRINT TABXY(22,7);" 2 .... Constant velocity stacked panels (CVS)"
2520  PRINT TABXY(22,9);" 3 .... CMP gather "
2530  PRINT TABXY(22,11);" 4 .... CMP stacked section"
2540  PRINT TABXY(22,13);" 5 .... RETURN to main menu"
2550  PRINT TABXY(19,Pointy);Marker$;
2560  Max_pointy=13
2570  RETURN
2580  !-------------------------------------------------------------------------
2590 Process_menu:!
2600  OUTPUT KBD;"K";
2610  PRINT TABXY(31,2);CHR$(129)&" Processing menu "&CHR$(128)
2620  PRINT TABXY(22,5);" 1 ..... Change polarity"
2630  PRINT TABXY(22,7);" 2 ..... Amplitude corrections to traces"
2640  PRINT TABXY(22,9);" 3 ..... Calculate static corrections"
2650  PRINT TABXY(22,11);" 4 ..... Velocity analysis"
2660  PRINT TABXY(22,13);" 5 ..... NMO correction and CMP stack "
2670  PRINT TABXY(22,15);" 6 ..... Filter traces"
2680  PRINT TABXY(22,17);" 7 ..... RETURN to main menu"
2690  PRINT TABXY(19,Pointy);Marker$;
2700  Max_pointy=17
2710  RETURN
2720  !-------------------------------------------------------------------------
2730 Display_menu:!
2740  OUTPUT KBD;"K";
2750  PRINT TABXY(35,2);CHR$(129)&" Display menu "&CHR$(128)
2760  PRINT TABXY(22,5);" 1 ..... To screen"
2770  PRINT TABXY(22,7);" 2 ..... To HP printer (normal graphics mode)"
2780  PRINT TABXY(22,9);" 3 ..... To HP printer (expanded graphics mode)"
2790  PRINT TABXY(22,11);" 4 ..... To plotter"
2800  PRINT TABXY(22,13);" 5 ..... RETURN to main menu"
2810  PRINT TABXY(19,Pointy);Marker$;
2820  Max_pointy=13
2830  RETURN
2840  !-------------------------------------------------------------------------
2850 Interpret_menu:!
2860  OUTPUT KBD;"K";
2870  PRINT TABXY(31,2);CHR$(129)&" Interpretation menu "&CHR$(128)
2880  PRINT TABXY(22,5);" 1 ..... Generate synthetic CMP gathers"
2890  PRINT TABXY(22,7);" 2 ..... Ray path modelling"
2900  PRINT TABXY(22,9);" 3 ..... Amplitude spectrum calculation"
2910  PRINT TABXY(22,11);" 4 ..... RETURN to main menu"
2920  PRINT TABXY(19,Pointy);Marker$;
2930  Max_pointy=11
2940  RETURN
2950
```

```
2960  !=============================================================================
2970 Screen_test: !
2980  Screen_test=1
2990  IF MAX(Screen_2(*))=0 THEN
3000    BEEP
3010    PRINT TABXY(5,15);"There is nothing to dump  DUMMY  -- Draw to screen be
fore trying to dump"
3020    BEEP
3030    WAIT 2.8
3040    BEEP
3050    Screen_test=0
3060  END IF
3070  RETURN
3080  !=============================================================================
3090  !
3100  END
```

```
10    !   BISON_DUMP
20    !   Dumps 24 channels from BISON to disc (unsorted)
30    !   Requires BASIC 3.0 with SERIAL BIN file
40    !
50    !   A.T.Dippenaar   ---   U.P.   ---    September 1986
60    !
70    !   Last changes on 01 October 1986
80    ! .
90    !=============================================================================
100   !
110   OPTION BASE 1
120   INTEGER Mixed_array(23280)
130   !
140   OUTPUT KBD;"K";
150   INPUT "Area identification ",Area$
160   INPUT "Line identification ",Line$
170   LOOP
180     INPUT "Shotpoint",Shot
190     IF Shot<0 THEN
200       Fil$=Area$&"_"&Line$&"n"&VAL$(ABS(Shot))&"M"
210     ELSE
220       Fil$=Area$&"_"&Line$&"_"&VAL$(Shot)&"M"
230     END IF
240     OUTPUT KBD;"K";
250   !
260     Sc=9                                  !  Select code for serial interface
270     ASSIGN @Bison TO 9
280   !
290     ON ERROR GOSUB Error
300     CONTROL Sc,0;1                        !  Resets interface
310     CONTROL Sc,3;9600                     !  Baud rate = 9600
320     CONTROL Sc,4;3+0+0                    !  8 bits/char, no parity, 1 stop bit
330     CONTROL Sc,5;3
340   !
350     DISP " Press <WRITE> <7> <ENTER> on Bison  --  then <CONTINUE> on HP9816
 "
360     PAUSE
370     DISP "
 "
380     PRINT TABXY(16,13);" Transfering data for file ";Fil$
390     CONTROL 2,7;1                         !  Locks out keyboard
400   !
410     OUTPUT @Bison;CHR$(27)&CHR$(65);      !  ESC A   ---   16 bit data
420     OUTPUT @Bison;CHR$(27)&CHR$(69);      !  ESC E   ---   Enhance memory
430     OUTPUT @Bison;CHR$(27)&CHR$(0);       !  ESC 0   ---   All the channels
440     DISP "   Please be patient  --  This will take just under 3  minutes  --
 Thank you. "
450   !
460     OUTPUT @Bison;CHR$(27)&CHR$(71);      !  ESC G   ---   start dump
470     ENTER @Bison USING "960(#,K),#,B,K,B,3(K),B,B,2(K)";Mixed_array(*)
480     OUTPUT @Bison;CHR$(27)&CHR$(81);      !  ESC Q   ---   Quit dump
490   !
500     DISP "
    "
510     ASSIGN @Bison TO *
520     CONTROL Sc,0;1                        !  Resets interface
530     CONTROL Sc,5;0
540     CONTROL 2,7;0                         !  Keyboard active again
550   !
560     BEEP
```

```
570
580     PRINT TABXY(16,13);"    Storing file ";Fil$;"
590     MASS STORAGE IS ":,700,1"
600     !
610     CREATE BDAT Fil$,24,1950
620     ASSIGN @To_disc TO Fil$
630     OUTPUT @To_disc;Mixed_array(*)
640     ASSIGN @To_disc TO *
650     !
660     MASS STORAGE IS ":,700"
670     PRINT TABXY(16,13);"   Finished with file ";Fil$;"
680     BEEP
690     OFF ERROR
700     DISP "                    Press <CONTINUE> to carry on"
710     PAUSE
720     DISP "
730   END LOOP
740   STOP
750   !=============================================================================
760 Error:  !
770   SELECT ERRN
780   CASE 167                                  !  Interface status error
790     STATUS 9,10;Uart_error
800     IF BIT(Uart_error,1) THEN Overrun
810     IF BIT(Uart_error,2) THEN Parity
820     IF BIT(Uart_error,3) THEN Framing
830     IF BIT(Uart_error,4) THEN Break
840   CASE 64                                   !  Mass Storage overflow
850     OUTPUT KBD;"K";
860     BEEP
870     PRINT TABXY(17,7);"*****************************************************"
880     PRINT TABXY(17,8);"*                                                   *"
890     PRINT TABXY(17,9);"*        Data disc full  --  Insert new disc        *"
900     PRINT TABXY(17,10);"*                                                  *
"
910     PRINT TABXY(17,11);"*****************************************************
"
920     BEEP
930     PAUSE
940     OUTPUT KBD;"K";
950   CASE 80                                   !  No disc in drive
960     OUTPUT KBD;"K";
970     BEEP
980     PRINT TABXY(17,7);"*****************************************************"
990     PRINT TABXY(17,8);"*                                                   *"
1000    PRINT TABXY(17,9);"*         No disc in drive --  Insert disc          *"
1010    PRINT TABXY(17,10);"*                                                  *
"
1020    PRINT TABXY(17,11);"*****************************************************
"
1030    BEEP
1040    PAUSE
1050    OUTPUT KBD;"K";
1060  CASE ELSE
1070    PRINT ERRN
1080  END SELECT
1090  GOTO 1190
1100  !
1110 Overrun:PRINT "OVERRUN"
1120  GOTO 1190
```

```
1130 Parity:PRINT "PARITY"
1140  GOTO 1190
1150 Framing:PRINT "FRAMING"
1160  GOTO 1190
1170 Break:PRINT "BREAK"
1180  !
1190  RETURN
1200  !=============================================================================
1210  END
```

```
10    !   BISON_SORT  --  Sorts mixed files into trace files
20    !
30    !   A.T.Dippenaar   ---   U.P.   ---    September 1986
40    !
50    !   Last changes on 26 April 1987
60    !=============================================================================
70    OPTION BASE 1
80    INTEGER Mixed_array(23280),Store_array(959)
90    !
100   PLOTTER IS 3,"INTERNAL"
110   X=100*MAX(1,RATIO)
120   Y=100*MAX(1,1/RATIO)
130   GINIT
140   GCLEAR
150   GRAPHICS ON
160   OUTPUT KBD;"K";
170   !
180   INPUT "Area identification (Maximum 2 character)",Area$
190   INPUT "Line identification (Maximum 3 characters)",Line$
200   INPUT "How many shotpoints",Amt_shot
210   INPUT "Shotpoint 1 location (Maximum 4 characters)",Shot_1
220   INPUT "Geophone 1 location",Geo_1
230   Geo_space=5
240   INPUT "Geophone spacing [ DEF = 5 ]",Geo_space
250   Shot_space=5
260   INPUT "Shotpoint spacing [ DEF = 5 ]",Shot_space
270   Graph_ques$="N"
280   INPUT "Graphic display of data before writing to disc (Y/N) [ DEF = N ]",G
raph_ques$
290   !
300   Last=Shot_1+((Amt_shot-1)*Shot_space)
310   FOR Shot=Shot_1 TO Last STEP Shot_space
320     MASS STORAGE IS ":,700,0"
330     IF Shot<0 THEN
340       Fil$=Area$&"_"&Line$&"n"&VAL$(ABS(Shot))&"M"
350     ELSE
360       Fil$=Area$&"_"&Line$&"_"&VAL$(Shot)&"M"
370     END IF
380     DISP "Reading file ";Fil$
390     ASSIGN @From_disc TO Fil$
400     ENTER @From_disc;Mixed_array(*)
410     ASSIGN @From_disc TO *
420     !
430     IF Graph_ques$="N" THEN 590
440     !
450     FOR Tr=1 TO 24
460       VIEWPORT (Tr-1)/27*X,(Tr+2)/27*X,28,100
470       WINDOW -35000,35000,-959,50
480       MOVE 0,10
490       LORG 4
500       CSIZE 3
510       LABEL VAL$(Tr)
520       MOVE Mixed_array((Tr-1)*970+1),0
530       FOR I=1 TO 959
540         DRAW Mixed_array((Tr-1)*970+I),-I
550       NEXT I
560     NEXT Tr
570     !
580     MASS STORAGE IS ":,700,1"
590     Counter=0
```

```
600     FOR Tr=1 TO 24
610     !
620       FOR I=1 TO 24
630         IF Tr=Ch_ignore(I) THEN 850
640       NEXT I
650       !
660       Counter=Counter+1
670       Ch_name$=VAL$(Geo_1+(Counter-1)*Geo_space)
680       !
690       IF Shot<0 THEN
700         Fil$=Line$&"n"&VAL$(ABS(Shot))&"_"&Ch_name$
710       ELSE
720         Fil$=Line$&"_"&VAL$(Shot)&"_"&Ch_name$
730       END IF
740       DISP "Storing file ";Fil$
750       FOR I=1 TO 959
760         Store_array(I)=Mixed_array((Tr-1)*970+I)      ! To save writing time
770       NEXT I
780       CREATE BDAT Fil$,1,1950
790       ASSIGN @To_disc TO Fil$
800       OUTPUT @To_disc;Store_array(*)
810       OUTPUT @To_disc;Mixed_array((Tr-1)*970+966)    ! Gain
820       OUTPUT @To_disc;Mixed_array((Tr-1)*970+969)    ! Sweep length
830       ASSIGN @To_disc TO *
840       !
850     NEXT Tr
860   NEXT Shot
870   MASS STORAGE IS ":,700"
880   DISP "
890   BEEP
900   END
```

```
10    !   COPY_SEIS  --  Copies seismic traces from field disc to hard disc
20    !
30    !   A.T.Dippenaar   ---   U.P.  ---   April 1987
40    !
50    !   Last changes on 02 May 1987
60    !=============================================================================
70    !
80    OPTION BASE 1
90    !
100   DIM Geo(24)
110   !=============================================================================
120   CONTROL 1,12;1
130   OUTPUT KBD;"K";
140   !
150   INPUT "Line identification (Maximum 2 alphanumericals)",Line$
160   MASS STORAGE IS ":,700,1"
170   DISP "Any missed shots on line ";Line$;
180   INPUT Miss_ques$
190   IF Miss_ques$="Y" THEN
200     INPUT "How many",Nr_missed_shots
210     ALLOCATE Missed_shot(Nr_missed_shots)
220     FOR I=1 TO Nr_missed_shots
230       DISP "Shotpoint of missed shot ";I;
240       INPUT Missed_shot(I)
250     NEXT I
260   END IF
270   !
280   Geo_space=5                 ! -
290   Shot_space=5                ! --  Default values
300   Shotpoint1=0                ! -
310   !
320   INPUT "First shotpoint location [ DEF = 0 ]",Shotpoint1
330   INPUT "Last shotpoint location",Shotpoint2
340   INPUT "First geophone position",Geo_1
350   !
360   INPUT "Shotpoint spacing [ DEF = 5 ]",Shot_space
370   INPUT "Geophone spacing [ DEF = 5 ]",Geo_space
380   Amt=12
390   INPUT "How many traces  per shot [ DEF = 12 ]",Amt
400   IF Amt>24 THEN
410     BEEP
420     DISP CHR$(129)&" You must be new around here  --  Max = 24 (wait) "&CHR$
(128)
430     WAIT 2
440     BEEP
450     GOTO 390
460   END IF
470     !
480   Sector=2
490   FOR Shot=Shotpoint1 TO Shotpoint2 STEP Shot_space
500     FOR M=1 TO Nr_missed_shots
510       IF Shot=Missed_shot(M) THEN 570
520     NEXT M
530     FOR Tr=1 TO Amt
540       Geo(Tr)=Geo_1+(Tr-1)*Geo_space
550       GOSUB Copy_trace
560     NEXT Tr
570   NEXT Shot
580     !
590   !=============================================================================
```

```
600   MASS STORAGE IS ":,700,0,6"
610   BEEP
620   Pointy=9
630   LOAD "AUTOST:,700,0,6"
640   !
650   STOP
660 !*************************************************************************
670 Copy_trace: !
680   !
690   IF Shot<0 THEN
700     Fil$=Line$&"n"&VAL$(ABS(Shot))&"_"&VAL$(Geo(Tr))
710   ELSE
720     Fil$=Line$&"_"&VAL$(Shot)&"_"&VAL$(Geo(Tr))
730   END IF
740   !
750   ON ERROR GOSUB Warning
760   COPY Fil$&":,700,1" TO Fil$&":,700,0,"&VAL$(Sector)
770   PRINT TABXY(20,18);"Copying trace ";Tr;" of ";Amt;"of shot ";Shot
780   !
790   !=========================================================================
800   ASSIGN @To_disc TO *
810   OFF ERROR
820   !
830   DISP "
840   RETURN
850   !-------------------------------------------------------------------------
860 Warning:  !
870   GRAPHICS OFF
880   DISP "
890   !
900   PRINT TABXY(10,7);"*******************************************************"
910   PRINT TABXY(10,8);"*                                                     *"
920   PRINT TABXY(10,10);"*                                                     *"
930   PRINT TABXY(10,11);"*******************************************************"
940   !
950   BEEP
960   SELECT ERRN
970   CASE 56
980     PRINT TABXY(10,9);"*             File not found on ,700,1
"
990     BEEP
1000  CASE 64
1010    IF Sector=2 THEN
1020      PRINT TABXY(10,9);"*              Switching to ,700,0,3
 *"
1030      Sector=3
1040      MASS STORAGE IS ":,700,0,3"
1050      GOTO 1150
1060    ELSE
1070      PRINT TABXY(10,9);"*    You have a problem ,700,0,3 is full as well
 *"
1080    END IF
1090  CASE 80
1100    PRINT TABXY(10,9);"*               No disc in :,700,1
*"
1110  CASE ELSE
1120    PRINT TABXY(10,9);"*      Unexpected error -- Error number = "&VAL$(ERRN
)&"         *"
1130  END SELECT
1140  PAUSE
```

```
1150  BEEP
1160  RETURN
1170  !-------------------------------------------------------------------------
1180  !
1190  END
```

```
10    !   READ_SEIS  --  Reads seismic traces from disc
20    !
30    !   A.T.Dippenaar   ---   U.P.  ---   September 1986
40    !
50    !   Last changes on 02 January 1988
60    !=============================================================================
70    !
80    OPTION BASE 1
90    !
100   COM /Seis_int/ Line$[2],INTEGER Trace(24,960),Opt,Amt,Gain(24)
110   COM /Seis_real/ REAL Shot(24),Geo(24),Shift_value(24),Offset
120   COM /Processed/ REAL Processed_trace(24,959),Pointy,V
130   COM /Polar/ INTEGER Pol_factor(24)
140   COM /Statics/ REAL Geophone_static(100),Source_static(100),Peg(100)
150   COM /Screens/ INTEGER Screen_1(7500),Screen_2(7500),REAL X,Y,Gain_amplitud
e,Gain_time
160   !
170   !=============================================================================
180   INTEGER Dummy(961),Sweep(24)
190   DIM Dum(959)
200   CONTROL 1,12;1
210   OUTPUT KBD;"K";
220   !
230   INPUT "Line identification (Maximum 2 alphanumericals)",Line$
240   IF Opt<4 THEN
250     Stat_ques$="Y"
260     INPUT "Must static calculations be read from disc [ DEF = Y ]",Stat_ques
$
270     IF MAX(Geophone_static(*))>0 AND Stat_ques$="N" THEN
280       INPUT "Do you wish to zero the static correction arrays (Y/N)",Sq$
290       IF Sq$="Y" THEN
300         FOR S=1 TO 100
310           Geophone_static(S)=0
320           Source_static(S)=0
330           Peg(S)=0
340         NEXT S
350         FOR S=1 TO 24
360           Shift_value(S)=0
370         NEXT S
380       ELSE
390         IF Sq$<>"N" THEN GOTO 280
400       END IF
410     END IF
420   ELSE
430     Stat_ques$="N"
440   END IF
450   !
460   IF Stat_ques$="Y" THEN
470     DISP "First peg position on line ";Line$;
480     INPUT First_peg
490     DISP "Last peg position on line ";Line$;
500     INPUT Last_peg
510     Peg_space=5
520     INPUT "Peg_spacing [ DEF = 5 ]",Peg_space
530     Peg_amt=(Last_peg-First_peg)/Peg_space+1
540   END IF
550   !
560   Format$="I"
570   INPUT "Data in  INTEGER (I) or REAL (R) format [ DEF = I ]",Format$
580   IF Format$<>"I" AND Format$<>"R" THEN 570
```

```
590   MASS STORAGE IS ":,700,1"
600   !
610   FOR Ch=1 TO 24
620     Pol_factor(Ch)=1
630   NEXT Ch
640   !
650   Geo_space=5                   ! -
660   Shot_space=5                  ! --  Default values
670   Cdp_fold=6                    ! -
680   !
690   SELECT Opt
700   !=============================================================================
710   !       Option 1  --  Field record
720   !=============================================================================
730   CASE 1
740   !
750     INPUT "Shot location",Shotpoint
760     INPUT "Geophone 1 location",Geo_1
770     INPUT "Geophone spacing [ DEF = 5 ]",Geo_space
780     Amt=24
790     INPUT "How many traces [ DEF = 24 ]",Amt
800     IF Amt>24 THEN
810       BEEP
820       DISP CHR$(129)&" You must be new around here  --  Max = 24 (wait) "&CH
R$(128)
830       WAIT 2
840       BEEP
850       GOTO 790
860     END IF
870     !
880     FOR Tr=1 TO Amt
890       Shot(Tr)=Shotpoint
900       Geo(Tr)=Geo_1+(Tr-1)*Geo_space
910       GOSUB Read_trace
920     NEXT Tr
930     !
940   !=============================================================================
950   !       Option 2  --  Uncorrected CDP traces
960   !=============================================================================
970   CASE 2
980     !
990     INPUT "Fold of proposed CDP stack [ DEF = 6 ] ( Any missed shots must be
 included )",Cdp_fold
1000    INPUT "Shot spacing [ DEF = 5 ]",Shot_space
1010    Amt=Cdp_fold
1020    INPUT "Midpoint of CMP gather",Cdp_centre
1030    Miss_ques$="N"
1040    INPUT "Any missed shots in this CMP gather (Y/N) [ DEF : N ]",Miss_ques$
1050    IF Miss_ques$="Y" THEN
1060      INPUT " How many ",Miss_amt
1070      ALLOCATE Missed(Miss_amt)
1080      INPUT " Enter as shot1,shot2,.....",Missed(*)
1090    END IF
1100    FOR Tr=1 TO Amt
1110      Shot(Tr)=Cdp_centre-((Tr-1)*Shot_space)
1120      FOR M=1 TO Miss_amt
1130        IF Shot(Tr)=Missed(M) THEN 1170
1140      NEXT M
1150      Geo(Tr)=Cdp_centre+((Tr-1)*Shot_space)
1160      GOSUB Read_trace
```

```
1170    NEXT Tr
1180  !=============================================================================
1190  !       Option 3  --  Common offset section
1200  !=============================================================================
1210  CASE 3
1220    !
1230    INPUT "First shotpoint for this section",First_shot
1240    INPUT "Last shotpoint for this section",Last_shot
1250    INPUT "Shot spacing [ DEF = 5 ]",Shot_space
1260    Amt=(Last_shot-First_shot)/Shot_space+1
1270    Miss_ques$="N"
1280    INPUT "Any missed shots in this section (Y/N) [ DEF : N ]",Miss_ques$
1290    IF Miss_ques$="Y" THEN
1300      INPUT " How many ",Miss_amt
1310      ALLOCATE Missed(Miss_amt)
1320      INPUT " Enter as shot1,shot2,.....",Missed(*)
1330    END IF
1340    INPUT "Offset",Offset
1350    !
1360    FOR Tr=1 TO Amt
1370      Shot(Tr)=First_shot+(Tr-1)*Shot_space
1380      FOR M=1 TO Miss_amt
1390        IF Shot(Tr)=Missed(M) THEN 1430
1400      NEXT M
1410      Geo(Tr)=Shot(Tr)+Offset
1420      GOSUB Read_trace
1430    NEXT Tr
1440    !
1450  !=============================================================================
1460  !       Option 4  --  CVG traces
1470  !=============================================================================
1480  CASE 4
1490    !
1500    MASS STORAGE IS ":,700,1"
1510    INPUT "What velocity",V
1520    INPUT "Fold of CDP gather [ DEF = 6 ]",Cdp_fold
1530    INPUT "Shot spacing [ DEF = 5 ]",Shot_space
1540    Amt=Cdp_fold
1550    INPUT "Shotpoint location of zero offset trace (midpoint of gather)",Cdp
_centre
1560    FOR Tr=1 TO Amt
1570      Shot(Tr)=Cdp_centre-((Tr-1)*Shot_space)
1580      Geo(Tr)=Cdp_centre+((Tr-1)*Shot_space)
1590      !
1600      Fil$=VAL$(Shot(Tr))&"v"&VAL$(V/100)&"g"&VAL$(Geo(Tr))
1610      !
1620      GOSUB Read_trace
1630    NEXT Tr
1640  !=============================================================================
1650  !       Option 5  --  Processed traces
1660  !=============================================================================
1670  CASE 5
1680    !
1690    MASS STORAGE IS ":,700,1"
1700    INPUT "How many midpoints",Nr_midpoints
1710    Amt=Nr_midpoints
1720    INPUT "First CDP position",First_cdp
1730    INPUT "Shot spacing [ DEF = 5 ]",Shot_space
1740    INPUT "Enter velocity ",V
1750    Shot(1)=First_cdp
```

```
1760    Shot(Amt)=First_cdp+(Amt-1)*Shot_space
1770    FOR Tr=1 TO Amt
1780      Mid=First_cdp+(Tr-1)*Shot_space
1790      !
1800      Fil$=VAL$(V)&"cvs"&VAL$(Mid)
1810      !
1820      GOSUB Read_trace
1830    NEXT Tr
1840  !=============================================================================
1850  !       Option 6  --  NMO corrected CMP gather
1860  !=============================================================================
1870  CASE 6
1880    INPUT "Midpoint of NMO corrected CMP gather",Cdp_centre
1890    INPUT "Shot spacing [ DEF = 5 ]",Shot_space
1900    FOR Tr=1 TO Amt
1910      Shot(Tr)=Cdp_centre-((Tr-1)*Shot_space)
1920      Geo(Tr)=Cdp_centre+((Tr-1)*Shot_space)
1930      !
1940      Fil$=VAL$(Shot(Tr))&"nmo"&VAL$(Geo(Tr))
1950      !
1960      GOSUB Read_trace
1970    NEXT Tr
1980    !
1990  !=============================================================================
2000  !       Option 7  --  NMO corrected CMP stacked section
2010  !=============================================================================
2020  CASE 7
2030    MASS STORAGE IS ":,700,1"
2040    INPUT "First CMP ",First_cmp
2050    INPUT "Last CMP ",Last_cmp
2060    Trace_space=5
2070    INPUT "Trace spacing [ DEF = 5 ]",Trace_space
2080    Amt=(Last_cmp-First_cmp)/Trace_space+1
2090    Shot(1)=First_cmp
2100    Shot(Amt)=First_cmp+(Amt-1)*Trace_space
2110    FOR Tr=1 TO Amt
2120      Position=First_cmp+(Tr-1)*Trace_space
2130      Fil$=Line$&"_"&VAL$(Position)
2140      GOSUB Read_trace
2150    NEXT Tr
2160  END SELECT
2170  !=============================================================================
2180  !         Static calculations read in
2190  !=============================================================================
2200  IF Stat_ques$="Y" THEN
2210    !
2220    MASS STORAGE IS ":,700,1"
2230    DISP " Reading static corrections "
2240    !
2250    F$=Line$&"_srce_st"
2260    ASSIGN @P TO F$
2270    ON END @P GOTO 2290
2280    ENTER @P;Source_static(*)
2290    ASSIGN @P TO *
2300    !
2310    F$=Line$&"_geo_st"
2320    ASSIGN @P TO F$
2330    ON END @P GOTO 2350
2340    ENTER @P;Geophone_static(*)
2350    ASSIGN @P TO *
```

```
2360    !
2370    FOR P=1 TO Peg_amt
2380      Peg(P)=First_peg+(P-1)*Peg_space
2390    NEXT P
2400    !
2410  ELSE
2420    FOR P=1 TO 100
2430      Peg(P)=0
2440    NEXT P
2450  END IF
2460  !===========================================================================
2470  MASS STORAGE IS ":,700,1"
2480  Pointy=9
2490  LOAD "AUTOST:,700,0"
2500  !
2510  STOP
2520!*****************************************************************************
2530 Read_trace:!
2540  !
2550  SELECT Opt
2560  CASE 1,2,3
2570    IF Shot(Tr)<0 THEN
2580      Fil$=Line$&"n"&VAL$(ABS(Shot(Tr)))&"_"&VAL$(Geo(Tr))
2590    ELSE
2600      Fil$=Line$&"_"&VAL$(Shot(Tr))&"_"&VAL$(Geo(Tr))
2610    END IF
2620  END SELECT
2630  !
2640  ON ERROR GOSUB Warning
2650  ASSIGN @To_disc TO Fil$
2660  ON END @To_disc GOTO 2760
2670  DISP "Reading trace ";Tr;" of ";Amt;"     ---    ";Fil$
2680  !
2690  IF Format$="I" THEN
2700    SELECT Opt
2710  !---------------------------------------------------------------------------
2720  !   INTEGER format      ---   Field records
2730  !---------------------------------------------------------------------------
2740    CASE 1,2,3
2750      ENTER @To_disc;Dummy(*)
2760      FOR I=1 TO 959
2770        Trace(Tr,I)=Dummy(I)
2780      NEXT I
2790      Gain(Tr)=Dummy(960)
2800      Sweep(Tr)=Dummy(961)
2810  !---------------------------------------------------------------------------
2820  !   INTEGER format      ---   Processed traces
2830  !---------------------------------------------------------------------------
2840    CASE 4,5,6,7
2850      ENTER @To_disc;Dummy(*)
2860      FOR I=1 TO 959
2870        Trace(Tr,I)=Dummy(I)
2880      NEXT I
2890    !
2900    END SELECT
2910  ELSE
2920    ENTER @To_disc;Dum(*)
2930    FOR I=1 TO 959
2940      Processed_trace(Tr,I)=Dum(I)
2950    NEXT I
```

```
2960    !
2970  END IF
2980  !=============================================================================
2990  ASSIGN @To_disc TO *
3000  OFF ERROR
3010  !
3020  RETURN
3030  !-----------------------------------------------------------------------------
3040 Warning: !
3050  GRAPHICS OFF
3060  DISP "
3070  !
3080  PRINT TABXY(10,7);"*********************************************************"
3090  PRINT TABXY(10,8);"*                                                         *"
3100  PRINT TABXY(10,10);"*                                                         *"
3110  PRINT TABXY(10,11);"*********************************************************"
3120  !
3130  BEEP
3140  SELECT ERRN
3150  CASE 56
3160    PRINT TABXY(10,9);"*         File not found on current MSU
"
3170    Msu_ques$="N"
3180    INPUT "Do you want to change MSU [ DEF = N ]",Msu_ques$
3190    IF Msu_ques$<>"Y" AND Msu_ques$<>"N" THEN 3180
3200    IF Msu_ques$="Y" THEN
3210      INPUT "Enter new MSU e.g. '':,700,0,3'' [ NO DEF ] ",Ms$
3220      ON ERROR GOTO 3250
3230      MASS STORAGE IS Ms$
3240      GOTO 3440
3250      OFF ERROR
3260      !
3270      SELECT ERRN
3280      CASE 150,52
3290        BEEP
3300        DISP "TRY AGAIN ---";
3310        GOTO 3210
3320      END SELECT
3330      !
3340    ELSE
3350      DISP "Insert next disc in current MSU   <CONTINUE>"
3360      PAUSE
3370    END IF
3380  CASE 80
3390    PRINT TABXY(10,9);"*           No disc in right hand drive
"
3400    PAUSE
3410  CASE ELSE
3420    PRINT TABXY(10,9);"*      Unexpected error -- Error number = "&VAL$(ERRN
)&"         *"
3430    PAUSE
3440  END SELECT
3450  OUTPUT KBD;"K";
3460  RETURN
3470  !-----------------------------------------------------------------------------
3480  !
3490  END
```

```
10    !   DISP_SEIS  --  Displays seismic section on screen
20    !
30    !   A.T.Dippenaar   ---   U.P  ---   September 1986
40    !
50    !   Last changes on 01 January 1988
60    !=============================================================================
70    !
80    OPTION BASE 1
90    !
100   COM /Seis_int/ Line$[2],INTEGER Trace(24,960),Opt,Amt,Gain(24)
110   COM /Seis_real/ REAL Shot(24),Geo(24),Shift_value(24),Offset
120   COM /Processed/ REAL Processed_trace(24,959),Pointy,V
130   COM /Polar/ INTEGER Pol_factor(24)
140   COM /Statics/ REAL Geophone_static(100),Source_static(100),Peg(100)
150   COM /Screens/ INTEGER Screen_1(7500),Screen_2(7500),REAL X,Y,Gain_amplitud
e,Gain_time
160   !
170   !=============================================================================
180   DIM Total_static(100),Label$[80]
190   !
200   Qq$="I"
210   INPUT "INTEGER (I) or REAL (R) format [ DEF = I ]",Qq$
220   PLOTTER IS 3,"INTERNAL"
230   CONTROL 1,12;1
240   X=100*MAX(1,RATIO)
250   Y=100*MAX(1,1/RATIO)
260   GINIT
270   GCLEAR
280   Shift=0
290   GRAPHICS ON
300   Gain_amplitude=1
310   Gain_time=1
320   Shift_flag$="clear"
330   Screen_toggle=0
340   Sweep=192                                          !   in msec
350   !
360   OUTPUT KBD;"K";
370   Sample_rate=(Sweep/1000)/959                       !   in sec
380   !
390   Var_area$="ON"
400   !
410 Time_shift_calc:  !
420   IF MAX(Peg(*))=0 THEN 540
430   FOR K=1 TO Amt
440     Total_static(K)=0
450     FOR Pg=1 TO 100
460       IF Shot(K)=Peg(Pg) THEN Total_static(K)=Total_static(K)+Source_static(
Pg)
470       IF Geo(K)=Peg(Pg) THEN Total_static(K)=Total_static(K)+Geophone_static
(Pg)
480     NEXT Pg
490     !
500     Shift_value(K)=INT(Total_static(K)/Sample_rate)
510     !
520   NEXT K
530   !
540   GOSUB Main
550   !
560   STOP
570 !=============================================================================
```

```
580 Main:
590   CONTROL 1,12;0
600   SET ECHO 1000000,10000000                    ! Off screen
610   FOR K=0 TO 9
620     ON KEY K LABEL "              " GOTO Spin
630   NEXT K
640   ON KEY 3 LABEL "   GAIN --> " GOSUB Gain_expand
650   ON KEY 8 LABEL "   GAIN <-- " GOSUB Gain_contract
660   ON KEY 5 LABEL "   DEFAULT " GOSUB Default
670   ON KEY 1 LABEL "    GAIN - " GOSUB Gain_sub
680   ON KEY 6 LABEL "    GAIN + " GOSUB Gain_add
690   ON KEY 7 LABEL " DRAW TRACES" GOSUB Draw_traces
700   ON KEY 4 LABEL "    RETURN" GOTO Autost
710   ON KEY 9 LABEL " NEXT SCREEN" GOSUB Screen
720   IF Var_area$="ON" THEN
730     ON KEY 2 LABEL "  FILL ON " GOSUB Fill_pos
740   ELSE
750     ON KEY 2 LABEL "  FILL OFF" GOSUB Fill_pos
760   END IF
770 Spin:GOTO Spin
780   RETURN
790   !-------------------------------------------------------------------------
800 Autost:  !
810   GRAPHICS OFF
820   Pointy=11
830   LOAD "AUTOST"
840   !-------------------------------------------------------------------------
850 Draw_traces: !
860   GCLEAR
870   FOR Tr=1 TO Amt
880     IF Tr>24 THEN
890       IF Tr=25 THEN
900         GOSUB Timing
910         GSTORE Screen_1(*)
920         GCLEAR
930       END IF
940     END IF
950     GOSUB Viewport
960     CLIP OFF
970     MOVE 0,10
980     LORG 4
990     CSIZE 3
1000    LABEL VAL$(Tr)
1010    MOVE Trace(Tr,1)*Pol_factor(Tr),0
1020    Gain_factor=1
1030    FOR I=1 TO 959*Gain_time
1040      IF Var_area$="ON" THEN
1050        IF Qq$="R" THEN
1060          IF Processed_trace(Tr,I)*Pol_factor(Tr)>0 THEN MOVE 0,-(I-Shift_va
lue(Tr))
1070        ELSE
1080          IF Trace(Tr,I)*Pol_factor(Tr)>0 THEN MOVE 0,-(I-Shift_value(Tr))
1090        END IF
1100      END IF
1110      IF Qq$="R" THEN
1120        DRAW (Processed_trace(Tr,I)*Pol_factor(Tr))*Gain_factor,-(I-Shift_va
lue(Tr))
1130      ELSE
1140        DRAW (Trace(Tr,I)*Pol_factor(Tr))*Gain_factor,-(I-Shift_value(Tr))
1150      END IF
```

```
1160    NEXT I
1170    !
1180    IF Tr=Amt THEN
1190      GOSUB Timing
1200      GSTORE Screen_2(*)
1210    END IF
1220  NEXT Tr
1230  RETURN
1240  ! ------------------------------------------------------------------------
1250 Screen: !
1260  IF Screen_toggle=0 THEN
1270    GLOAD Screen_1(*)
1280    Screen_toggle=1
1290    Tr=1
1300  ELSE
1310    GLOAD Screen_2(*)
1320    Screen_toggle=0
1330    Tr=25
1340  END IF
1350  IF Shift_flag$="set" THEN
1360    MOVE -6,-Point
1370    DRAW Amt,-Point
1380  END IF
1390  RETURN
1400  !------------------------------------------------------------------------
1410 Timing: !
1420  VIEWPORT 1/30*X,27/30*X,28,100
1430  WINDOW 0,100,-959*Gain_time,50*Gain_time
1440  !
1450  CLIP OFF
1460  PEN -1
1470  FOR D=1 TO 50
1480    MOVE 0,D
1490    DRAW 105,D
1500  NEXT D
1510  PEN 1
1520  !
1530  LORG 8
1540  FOR Timing=0 TO Sweep STEP 50
1550    Position=Timing*959/Sweep
1560    MOVE 6,-Position
1570    CSIZE 3
1580    LABEL Timing
1590    LINE TYPE 4
1600    MOVE 6,-Position
1610    DRAW 100,-Position
1620    LINE TYPE 1
1630  NEXT Timing
1640  !
1650  SELECT Opt
1660  CASE 1
1670    Label$="LINE "&Line$&"   SHOTPOINT "&VAL$(Shot(1))&"  GEO 1 = "&VAL$(Geo
(1))&"  GEO"&VAL$(Amt)&" = "&VAL$(Geo(Amt))
1680  CASE 2
1690    Label$="LINE "&Line$&"   CMP GATHER "&VAL$(Shot(1))&"  GEO 1 = "&VAL$(Ge
o(1))&"  GEO"&VAL$(Amt)&" = "&VAL$(Geo(Amt))
1700  CASE 3
1710    Label$="LINE "&Line$&"   CONSTANT OFFSET "&VAL$(Offset)&"   SHOTPOINT "&
VAL$(Shot(1))&"  TO SHOTPOINT "&VAL$(Shot(Amt))
1720  CASE 4
```

```
1730    Label$="LINE "&Line$&"   CONSTANT VELOCITY GATHER :"&VAL$(V)&" M/S  CMP
"&VAL$(Shot(1))&" TO "&VAL$(Shot(Amt))
1740  CASE 5
1750    Label$="LINE "&Line$&"   CONSTANT VELOCITY STACKS :"&VAL$(V)&" M/S  CMP
"&VAL$(Shot(1))&" TO "&VAL$(Shot(Amt))
1760  CASE 6
1770    Label$="LINE "&Line$&"   CORRECTED CMP GATHER "&VAL$(Shot(1))&"  GEO 1 =
 "&VAL$(Geo(1))&"  GEO"&VAL$(Amt)&" = "&VAL$(Geo(Amt))
1780  CASE 7
1790    Label$="LINE "&Line$&" : NMO STACKED SECTION -- SHOTPOINT "&VAL$(Shot(1)
)&"  TO SHOTPOINT "&VAL$(Shot(Amt))
1800  END SELECT
1810  MOVE 50,8
1820  LORG 4
1830  CSIZE 3.3
1840  LABEL Label$
1850  RETURN
1860  !-------------------------------------------------------------------------
1870 Fill_pos:     !
1880  IF Var_area$="ON" THEN
1890    ON KEY 2 LABEL "  FILL OFF " GOSUB Fill_pos
1900    Var_area$="OFF"
1910  ELSE
1920    ON KEY 2 LABEL "  FILL ON " GOSUB Fill_pos
1930    Var_area$="ON"
1940  END IF
1950  RETURN
1960 !-------------------------------------------------------------------------
1970 Gain_add: !
1980  Gain_amplitude=Gain_amplitude/2
1990  GOSUB Gain_display
2000  RETURN
2010 !-------------------------------------------------------------------------
2020 Gain_sub:!
2030  Gain_amplitude=Gain_amplitude*2
2040  GOSUB Gain_display
2050  RETURN
2060  !-------------------------------------------------------------------------
2070 Gain_expand: !
2080  Gain_time=Gain_time/2
2090  GOSUB Gain_display
2100  RETURN
2110  !-------------------------------------------------------------------------
2120 Gain_contract: !
2130  Gain_time=Gain_time*2
2140  GOSUB Gain_display
2150  RETURN
2160  !-------------------------------------------------------------------------
2170 Gain_display: !
2180  DISP TAB(14);"Amplitude factor = ";1/Gain_amplitude;
2190  DISP TAB(46);"Expand factor = ";1/Gain_time
2200  RETURN
2210  !-------------------------------------------------------------------------
2220 Default: !
2230  Gain_time=1
2240  Gain_amplitude=1
2250  GOSUB Gain_display
2260  RETURN
2270  !-------------------------------------------------------------------------
2280 Viewport:!
```

```
2290  T=Tr
2300  IF Tr>24 THEN T=Tr-24
2310  Xxx=T+3
2320  VIEWPORT (Xxx-3)/30*X,(Xxx+3)/30*X,28,100
2330  Left=-128000*Gain_amplitude
2340  Right=128000*Gain_amplitude
2350  Bottom=(-959*Gain_time)
2360  Top=50*Gain_time
2370  WINDOW Left,Right,Bottom,Top
2380  RETURN
2390  !-------------------------------------------------------------------------
2400  END
```

```
10    !   PLOT_SEIS
20    !
30    !   A.T.Dippenaar   ---   U.P  ---   October 1986
40    !
50    !   Last changes on 02 May 1987
60    !=============================================================================
70    !
80    OPTION BASE 1
90    !
100   INTEGER Trace(100,959),Dummy(959),Pol_factor(100),Shift_value(100)
110   DIM Geophone_static(100),Source_static(100),Peg(100)
120   !
130   !=============================================================================
140   !
150   DIM Comment1$[80],Comment2$[80],Comment3$[80]
160   Gain_amplitude=1
170   Gain_time=1
180   FOR C=1 TO 100
190     Pol_factor(C)=1
200   NEXT C
210   !
220   OUTPUT KBD;"K";
230   !
240   INPUT "Stacked CMP section (CMP) or common offset section (COS)",Type_ques
$
250   IF Type_ques$<>"CMP" AND Type_ques$<>"COS" THEN 240
260   INPUT "Line identification (Maximum 1 alphanumerical )",Line$
270   !
280   OUTPUT KBD;"K";
290   !
300   INPUT "Paper size A3 (1) or A3 long (2)",Pap
310   IF Pap=1 THEN Paper$="A3"
320   IF Pap=2 THEN Paper$="A3_LONG"
330   !
340   INPUT "Client ",Client$
350   INPUT "Area description ",Area_descrip$
360   INPUT "Date of survey",Date$
370   INPUT "Title of plot",Head$
380   INPUT " Any comments that must be labeled (Y/N)",Ques$
390   IF Ques$="Y" THEN
400     INPUT "Comment 1 (max 80 characters)",Comment1$
410     INPUT "Comment 2 (max 80 characters)",Comment2$
420     INPUT "Comment 3 (max 80 characters)",Comment3$
430   END IF
440   !
450   MASS STORAGE IS ":,700,1"
460   !
470   SELECT Type_ques$
480   CASE "CMP"
490     INPUT "First CMP point ",First_cmp
500     INPUT "Last CMP point ",Last_cmp
510     INPUT "CMP spacing",Cmp_space
520     Amt=(Last_cmp-First_cmp)/Cmp_space+1
530     FOR Tr=1 TO Amt
540       Cmp=First_cmp+(Tr-1)*Cmp_space
550       Fil$=Line$&"_"&VAL$(Cmp)
560       GOSUB Read_trace
570     NEXT Tr
```

```
580   CASE "COS"
590     INPUT "First shotpoint ",First_shot
600     INPUT "Last shotpoint ",Last_shot
610     INPUT "Shot spacing",Shot_space
620     Amt=(Last_shot-First_shot)/Shot_space+1
630     INPUT "Offset",Offset
640     FOR Tr=1 TO Amt
650       Shot=First_shot+(Tr-1)*Shot_space
660       Fil$=Line$&"_"&VAL$(Shot)&"_"&VAL$(Shot+Offset)
670       GOSUB Read_trace
680     NEXT Tr
690   END SELECT
700   !
710   MASS STORAGE IS ":,700,0,6"
720   !
730   Var_area$="ON"
740   GOSUB Main
750   !
760   STOP
770 !=============================================================================
780 Read_trace: !
790   DISP "Reading trace ";Tr;" of ";Amt;"  ---  ";Fil$
800   ASSIGN @To_disc TO Fil$
810   ON END @To_disc GOTO 830
820   ENTER @To_disc;Dummy(*)
830   FOR I=1 TO 959
840     Trace(Tr,I)=Dummy(I)
850   NEXT I
860   ASSIGN @To_disc TO *
870   !
880   RETURN
890 !-----------------------------------------------------------------------------
900 Main:  !
910   CONTROL 1,12;0
920   FOR K=0 TO 9
930     ON KEY K LABEL "              " GOTO Spin
940   NEXT K
950   ON KEY 0 LABEL " PLOT TRACES" GOSUB Plot_traces
960   ON KEY 2 LABEL "   GAIN --> " GOSUB Gain_expand
970   ON KEY 7 LABEL "   GAIN <-- " GOSUB Gain_contract
980   ON KEY 4 LABEL "   DEFAULT " GOSUB Default
990   ON KEY 1 LABEL "    GAIN - " GOSUB Gain_sub
1000  ON KEY 6 LABEL "    GAIN + " GOSUB Gain_add
1010  ON KEY 8 LABEL " CHANGE POL" GOSUB Polarity
1020  IF Var_area$="ON" THEN
1030    ON KEY 9 LABEL "  FILL ON " GOSUB Fill_pos
1040  ELSE
1050    ON KEY 9 LABEL "  FILL OFF" GOSUB Fill_pos
1060  END IF
1070 Spin:GOTO Spin
1080  RETURN
1090  !-----------------------------------------------------------------------------
1100 Plot_traces:!
1110  Sweep=192
1120  PLOTTER IS 705,"HPGL"
1130  X=100*MAX(1,RATIO)
1140  Y=100*MAX(1,1/RATIO)
1150  !
1160  VIEWPORT 0,X,0,Y
1170  WINDOW 0,200,0,100
```

```
1180  LORG 3
1190  CSIZE 3.8
1200  FOR L=-2 TO 2
1210    MOVE 3+L/15,99
1220    LABEL "Line "&Line$
1230  NEXT L
1240  LORG 9
1250  FOR L=-2 TO 2
1260    MOVE 197+L/15,99
1270    LABEL "Line "&Line$
1280  NEXT L
1290  !
1300  IF Paper$="A3" THEN
1310    VIEWPORT 0,X,0,.25*Y
1320  ELSE
1330    VIEWPORT .5*X,X,0,.25*Y
1340  END IF
1350  WINDOW 0,100,0,25
1360  FRAME
1370  MOVE 0,25
1380  DRAW 100,25
1390  MOVE 70,0
1400  DRAW 70,25
1410  CSIZE 2
1420  LORG 8
1430  MOVE 80,20
1440  LABEL "Client : "
1450  MOVE 80,15
1460  LABEL "Area : "
1470  MOVE 80,10
1480  LABEL "Line : "
1490  MOVE 80,5
1500  LABEL "Date : "
1510  LORG 2
1520  MOVE 80,5
1530  LABEL Date$
1540  MOVE 80,10
1550  LABEL Line$
1560  MOVE 80,15
1570  LABEL Area_descrip$
1580  MOVE 80,20
1590  LABEL Client$
1600  !
1610  CSIZE 2.8
1620  LORG 6
1630  FOR L=-1 TO 1
1640    MOVE 35+L/12,25
1650    LABEL Head$
1660  NEXT L
1670  !
1680  CSIZE 2.2
1690  LORG 2
1700  MOVE 10,16
1710  LABEL "Sweep length : 192 msec"
1720  CSIZE 2
1730  MOVE 5,10
1740  LABEL Comment1$
1750  MOVE 5,6
1760  LABEL Comment2$
1770  MOVE 5,2
```

```
1780  LABEL Comment3$
1790  !
1800  ! ---------------------- Timing mark routine ----------------------------
1810  !
1820  VIEWPORT 0,.95*X,.28*Y,.94*Y
1830  WINDOW 0,100,-959*Gain_time,50*Gain_time
1840  LORG 8
1850  FOR Timing=0 TO Sweep STEP 50
1860    Position=Timing*959/Sweep
1870    MOVE 4,-Position
1880    CSIZE 2.5
1890    LABEL Timing
1900    IF Timing>=50 THEN LINE TYPE 4
1910    MOVE 4,-Position
1920    DRAW 100,-Position
1930    LINE TYPE 1
1940  NEXT Timing
1950  !
1960  !---------------------- Trace drawing routine ------------------------
1970  !
1980  FOR Tr=1 TO Amt
1990    GOSUB Viewport
2000    CLIP OFF
2010    MOVE 0,10
2020    LORG 4
2030    CSIZE 2
2040    LABEL VAL$(First_cmp+(Tr-1)*Cmp_space)
2050    MOVE Trace(Tr,1)*Pol_factor(Tr),0
2060    Gain_factor=1
2070    IF Opt=2 THEN Gain_factor=2^((Norm_gain-Gain(Tr))/6)
2080    FOR I=1 TO 959*Gain_time
2090      IF Var_area$="ON" THEN
2100        IF Trace(Tr,I)*Pol_factor(Tr)>0 THEN MOVE 0,-(I-Shift_value(Tr))
2110      END IF
2120      DRAW (Trace(Tr,I)*Pol_factor(Tr))*Gain_factor,-(I-Shift_value(Tr))
2130    NEXT I
2140    !
2150  NEXT Tr
2160  MASS STORAGE IS ":,700,0,6"
2170  LOAD "AUTOST"
2180  RETURN
2190  !-------------------------------------------------------------------------
2200 Fill_pos:     !
2210  IF Var_area$="ON" THEN
2220    ON KEY 9 LABEL "  FILL OFF " GOSUB Fill_pos
2230    Var_area$="OFF"
2240  ELSE
2250    ON KEY 9 LABEL "  FILL ON " GOSUB Fill_pos
2260    Var_area$="ON"
2270  END IF        !
2280  RETURN
2290 !
2300 Gain_add: !
2310  Gain_amplitude=Gain_amplitude/2
2320  PRINT TABXY(65,18);"Gain_factor=";1/Gain_amplitude
2330  RETURN
2340 !
2350 Gain_sub:!
2360  Gain_amplitude=Gain_amplitude*2
2370  PRINT TABXY(65,18);"Gain_factor=";1/Gain_amplitude
```

```
2380  RETURN
2390  !
2400 Gain_expand: !
2410  Gain_time=Gain_time/2
2420  RETURN
2430  !
2440 Gain_contract: !
2450  Gain_time=Gain_time*2
2460  RETURN
2470  !
2480 Default: !
2490  Gain_time=1
2500  Gain_amplitude=1
2510  RETURN
2520  !
2530 Polarity: !
2540  FOR C=1 TO 100
2550    Pol_factor(C)=Pol_factor(C)*(-1)
2560  NEXT C
2570  RETURN
2580   !
2590 Viewport:!
2600  Xxx=Tr+3
2610  IF Paper$="A3" THEN VIEWPORT (Xxx-3)/30*X,(Xxx+3)/30*X,.28*Y,.94*Y
2620  IF Paper$="A3_LONG" THEN VIEWPORT (Xxx-3)/70*X,(Xxx+3)/70*X,.28*Y,.94*Y
2630  WINDOW -64000*Gain_amplitude,64000*Gain_amplitude,-959*Gain_time,50*Gain_t
ime
2640  RETURN
2650 !
2660  END
```

```
10    !   STATICS  --  Calculates source and geophone statics using uphole times
20    !
30    !   A.T.Dippenaar   ---   U.P.  ---   January 1987
40    !
50    !   Last changes on 11 January 1988
60    !=============================================================================
70    !
80    OPTION BASE 1
90    !
100   INTEGER Trace(24,960),Dum(961)
110   !
120   !=============================================================================
130   !
140   MASS STORAGE IS ":,700,1  "
150   OUTPUT KBD;"K";
160   CONTROL 1,12;1
170   X=100*MAX(1,RATIO)
180   Y=100*MAX(1,1/RATIO)
190   GINIT
200   GCLEAR
210   GRAPHICS ON
220   Gain_amplitude=1
230   Gain_time=1
240   Sweep=192                                          !   in msec
250   Sample_rate=(Sweep/1000)/959                       !   in sec
260   !
270   !=============================================================================
280   !
290   OUTPUT KBD;"K";
300   INPUT "Line identifier ",Line$
310   INPUT "First shot position ",First_shot
320   INPUT "Last shot position ",Last_shot
330   INPUT "Last peg (geophone) position ",Last_peg
340   Peg_space=5
350   First_peg=First_shot
360   INPUT "Peg seperation [ DEF = 5 ]",Peg_space
370   Miss_ques$="N"
380   INPUT "Any missed shots [ DEF = N ] (Y/N)",Miss_ques$
390   IF Miss_ques$="Y" THEN
400     INPUT "How many",Miss_amt
410     IF Miss_amt=0 THEN 440
420     ALLOCATE Missed_shots(Miss_amt)
430     INPUT "Enter all the missed shots (shot1,shot2,shot3,.....)",Missed_shot
s(*)
440   END IF
450   Peg_amt=(Last_peg-First_shot)/Peg_space+1
460   Shot_amt=(Last_shot-First_shot)/Peg_space+1-Miss_amt
470   !
480   ALLOCATE Geophone_static(Peg_amt),Source_static(Peg_amt),Peg(Peg_amt)
490   !
500   Nr_sets=INT(Shot_amt/24)+1
510   !
520   FOR Pg=1 TO Peg_amt
530     Peg(Pg)=First_peg+((Pg-1)*Peg_space)
540   NEXT Pg
550   !
560   ALLOCATE Upper_velocity(Peg_amt),Uphole_time(Peg_amt),Shot_depth(Peg_amt),
Shot_height(Peg_amt),Height(Peg_amt)
570   !
580   FOR P=1 TO Peg_amt
```

```
590     DISP "Enter the true height at peg ";Peg(P);
600     INPUT Height(P)
610   NEXT P
620   !
630   OUTPUT KBD;"K";
640   PRINT TABXY(1,5);Nr_sets-1;" sets of 24 traces and 1 set of ";Shot_amt-(Nr
_sets-1)*24-Miss_amt;" zero offset traces will be displayed"
650   PRINT TABXY(1,8);"   Using the KNOB and the 'ENTER' key pick the first bre
ak on every trace"
660   PRINT TABXY(1,11);"After every set of 24 traces have been picked press the
 ' NEXT SET ' softkey "
670   !
680   !=============================================================================
690   !
700   INPUT "Were all the shot holes the same depth (Y/N)",Ques$
710   SELECT Ques$
720   CASE "Y"
730     INPUT "What depth",Depth
740     FOR P=1 TO Peg_amt
750       Shot_depth(P)=Depth
760     NEXT P
770   CASE "N"
780     FOR P=1 TO Peg_amt
790       DISP "Depth of shot hole at shotpoint (Enter 0 if no shot)";Peg(P);
800       INPUT Shot_depth(P)
810     NEXT P
820   CASE ELSE
830     BEEP
840     GOTO 700
850   END SELECT
860   !
870   !=============================================================================
880   !
890   FOR P=1 TO Peg_amt
900     Shot_height(P)=Height(P)-Shot_depth(P)
910   NEXT P
920   !
930   Datum_height=MIN(Shot_height(*))
940   !
950   !=============================================================================
960   !
970   !
980   Trace_counter=1
990   FOR Set=1 TO Nr_sets
1000    GOSUB Reader
1010    GOSUB Draw_traces
1020    Tr=1
1030    Point=1
1040    GOSUB Viewport
1050    SET ECHO Trace(Tr,1),0
1060    GOSUB Pick_break
1070    OFF KEY
1080    SET ECHO 10000000,1000000               !  Echo off screen
1090    GCLEAR
1100    OUTPUT KBD;"K";
1110  NEXT Set
1120  !
1130  GOSUB Stat_calc
1140  !
1150  STOP
```

```
1160  !-------------------------------------------------------------------------
1170 Reader: !
1180  IF Nr_sets=1 THEN
1190    Nr=Shot_amt
1200  ELSE
1210    IF Set<>Nr_sets THEN
1220      Nr=24
1230    ELSE
1240      Nr=Shot_amt MOD ((Nr_sets-1)*24)
1250    END IF
1260  END IF
1270  GCLEAR
1280  FOR Tr=1 TO Nr+(Nr_sets=Set)*Miss_amt
1290    Pos=First_peg+(((Set-1)*24)+Tr-1)*Peg_space
1300      !
1310    FOR M=1 TO Miss_amt
1320      IF Pos=Missed_shots(M) THEN 1470
1330    NEXT M
1340      !
1350    ON ERROR GOSUB To_err_is_human
1360    Fil$=Line$&"_"&VAL$(Pos)&"_"&VAL$(Pos)
1370    DISP TAB(30);"Reading ";Fil$
1380    ASSIGN @P TO Fil$
1390    ON END @P GOTO 1410
1400    ENTER @P;Dum(*)
1410    ASSIGN @P TO *
1420    OFF ERROR
1430      !
1440    FOR I=1 TO 959
1450      Trace(Tr,I)=Dum(I)
1460    NEXT I
1470  NEXT Tr
1480  !
1490  OUTPUT KBD;"K";
1500  RETURN
1510 !=========================================================================
1520 Pick_break:!
1530  !
1540  CONTROL 1,12;1
1550  ON KNOB .01 GOSUB Move_echo_pick
1560  ON KBD GOSUB Keyboard
1570  FOR K=0 TO 9
1580    ON KEY K LABEL " " GOTO Spinner
1590  NEXT K
1600  !
1610 Spinner:GOTO Spinner
1620  RETURN
1630  !=========================================================================
1640  !
1650 Move_echo_pick:
1660  SELECT KNOBX
1670  CASE >0
1680    Point=Point+1
1690  CASE ELSE
1700    Point=Point-1
1710  END SELECT
1720  IF Point<1 THEN Point=1
1730  IF Point>959 THEN Point=959
1740  PRINT TABXY(1,18);INT(Point/959*192);" ms
1750  SET ECHO Trace(Tr,Point),-Point
```

```
1760  RETURN
1770  !-------------------------------------------------------------------------
1780 Keyboard: !
1790  SELECT KBD$
1800  CASE CHR$(255)&"E",CHR$(255)&"X"&CHR$(255)&"C"
1810    Uphole_time(Trace_counter)=Point/959*192
1820    Posi=First_peg+(Trace_counter-1)*Peg_space
1830    FOR Mm=1 TO Miss_amt
1840      IF Posi=Missed_shot(M) THEN Posi=Posi+Peg_space
1850    NEXT Mm
1860    DISP TAB(1);"Trace ";Trace_counter;
1870    DISP TAB(15);" : Peg ";Posi;
1880    DISP TAB(30);" Uphole time = ";INT(Uphole_time(Trace_counter));" msec"
1890    GOSUB Next_trace
1900  CASE ELSE
1910    BEEP 1800,.15
1920  END SELECT
1930  RETURN
1940  !-------------------------------------------------------------------------
1950 Next_trace: !
1960  Tr=Tr+1
1970  Trace_counter=Trace_counter+1
1980  IF Trace_counter>Shot_amt THEN Stat_calc
1990  IF Tr>Nr THEN
2000    Tr=Nr
2010    OFF KBD
2020    CONTROL 1,12;0
2030    ON KEY 7 LABEL " NEXT SET " RECOVER 1070
2040    GOTO 2080
2050  END IF
2060  GOSUB Viewport
2070  SET ECHO Trace(Tr,Point),-Point
2080  RETURN
2090  !-------------------------------------------------------------------------
2100 Stat_calc:  !
2110    !
2120  Pp=0
2130  FOR P=1 TO Shot_amt+Miss_amt
2140    Pp=Pp+1
2150    FOR M=1 TO Miss_amt
2160      IF Peg(P)=Missed_shots(M) THEN
2170        Geophone_static(P)=Geophone_static(P-1)
2180        Source_static(P)=Source_static(P-1)
2190        Pp=Pp-1
2200        GOTO 2270
2210      END IF
2220    NEXT M
2230    Upper_velocity(P)=Shot_depth(P)/(Uphole_time(Pp)/1000)          ! In m/s
2240    IF Upper_velocity(P)=0 THEN 2270
2250    Geophone_static(P)=(Height(P)-Datum_height)/Upper_velocity(P)   ! In sec
2260    Source_static(P)=(Height(P)-Shot_depth(P)-Datum_height)/Upper_velocity(P
)
2270  NEXT P
2280  !
2290  FOR P=Shot_amt++Miss_amt+1 TO Peg_amt
2300    Geophone_static(P)=Geophone_static(P-1)
2310    Source_static(P)=Source_static(P-1)
2320  NEXT P
2330  !
2340  SET ECHO 1000000,1000000
```

```
2350  GRAPHICS OFF
2360  OUTPUT KBD;"K";
2370  !
2380  BEEP
2390  DISP "Statics disc in RH drive < CONTINUE > to carry on"
2400  PAUSE
2410  !
2420  MASS STORAGE IS ":,700,1  "
2430  Fil$=Line$&"_surface"
2440  CREATE BDAT Fil$,1,Peg_amt*16
2450  ASSIGN @P TO Fil$
2460  ON END @P GOTO 2490
2470  OUTPUT @P;Height(*)
2480  OUTPUT @P;Shot_depth(*)
2490  ASSIGN @P TO *
2500  !
2510  Fil$=Line$&"_geo_st"
2520  CREATE BDAT Fil$,1,Peg_amt*8
2530  ASSIGN @P TO Fil$
2540  ON END @P GOTO 2560
2550  OUTPUT @P;Geophone_static(*)
2560  ASSIGN @P TO *
2570  !
2580  Fil$=Line$&"_srce_st"
2590  CREATE BDAT Fil$,1,Peg_amt*8
2600  ASSIGN @P TO Fil$
2610  ON END @P GOTO 2630
2620  OUTPUT @P;Source_static(*)
2630  ASSIGN @P TO *
2640  !
2650  LOAD "AUTOST:,700,0  "
2660  !
2670  RETURN
2680  !
2690!=============================================================================
2700 Draw_traces:!
2710  GCLEAR
2720  FOR Tr=1 TO Nr
2730    GOSUB Viewport
2740    CLIP OFF
2750    MOVE 0,1
2760    LORG 4
2770    CSIZE 3
2780    LABEL VAL$(Tr)
2790    MOVE Trace(Tr,1),0
2800    FOR I=1 TO 959*Gain_time
2810      DRAW Trace(Tr,I),-I
2820    NEXT I
2830    !
2840    IF Tr=Nr THEN
2850      GOSUB Timing
2860    END IF
2870  NEXT Tr
2880  RETURN
2890  ! ---------------------------------------------------------------------
2900 Timing: !
2910  VIEWPORT 1/30*X,27/30*X,28,100
2920  WINDOW 0,100,-959*Gain_time,50*Gain_time
2930  !
2940  CLIP OFF
```

```
2950  !
2960  LORG 8
2970  FOR Timing=0 TO 40 STEP 5
2980    Position=Timing*959/Sweep
2990    MOVE 6,-Position
3000    CSIZE 3
3010    LABEL Timing
3020    LINE TYPE 4
3030    MOVE 6,-Position
3040    DRAW 100,-Position
3050    LINE TYPE 1
3060  NEXT Timing
3070  !
3080  RETURN
3090  !-------------------------------------------------------------------------
3100 Viewport:!
3110  Gain_time=.25
3120  Gain_amplitude=1
3130  Xxx=Tr+3
3140  VIEWPORT (Xxx-3)/30*X,(Xxx+3)/30*X,28,100
3150  Left=-128000*Gain_amplitude
3160  Right=128000*Gain_amplitude
3170  Bottom=(-959*Gain_time)
3180  Top=50*Gain_time
3190  WINDOW Left,Right,Bottom,Top
3200  RETURN
3210  !-------------------------------------------------------------------------
3220  !
3230 To_err_is_human: !
3240  IF ERRN=56 THEN
3250    BEEP
3260    OUTPUT KBD;"K";
3270    PRINT TABXY(20,14);" File ";Fil$;" not found on current MSU"
3280    Msu_ques$="N"
3290    INPUT "                  Do you want to change the MSU    [ DEF = N ]",M
su_ques$
3300    IF Msu_ques$="Y" THEN
3310      INPUT "Enter new MSUS  e.g.  '':,700,0,3'' (use double quotes)",Ms$
3320      ON ERROR GOTO Bad_mas_storage
3330      MASS STORAGE IS Ms$
3340      OFF ERROR
3350    ELSE
3360      DISP "                       Insert next data disc < CONTINUE >"
3370      PAUSE
3380    END IF
3390    OUTPUT KBD;"K";
3400  ELSE
3410    DISP " oops   ERRN = ";ERRN
3420    PAUSE
3430  END IF
3440  RETURN
3450  !
3460 Bad_mas_storage: !
3470  SELECT ERRN
3480  CASE 150,52
3490    BEEP
3500    DISP "TRY AGAIN --- ";
3510  CASE ELSE
3520    DISP " oops   ---  ERRN = ";ERRN
3530    PAUSE
```

```
3540  END SELECT
3550  GOTO 3310
3560  !=============================================================================
3570  END
```

```
10    !   POLAR  :  Change polarity of chosen channels
20    !
30    !   January 1987   ---   U.P.   ---   A.T.Dippenaar
40    !
50    !   Last changes on 02 May 1987
60    !=============================================================================
70    !
80    OPTION BASE 1
90    !
100   COM /Seis_int/ INTEGER Trace(24,960),Opt,Amt,Gain(24)
110   COM /Seis_real/ REAL Shot(24),Geo(24),Shift_value(24),Offset
120   COM /Processed/ REAL Processed_trace(24,959),Pointy,V
130   COM /Polar/ INTEGER Pol_factor(24)
140   COM /Statics/ REAL Geophone_static(100),Source_static(100),Peg(100)
150   COM /Screens/ INTEGER Screen_1(7500),Screen_2(7500),REAL X,Y,Gain_amplitud
e,Gain_time
160   !
170   !=============================================================================
180   INTEGER Pol_ch(24)
190   OUTPUT KBD;"K";
200   !
210   INPUT "Do you wish to change the polarity of all the channels (Y/N)",Ques$
220   IF Ques$="Y" THEN
230     FOR Ch=1 TO 24
240       Pol_factor(Ch)=Pol_factor(Ch)*(-1)
250     NEXT Ch
260   ELSE
270     INPUT "What channels",Pol_ch(*)
280     FOR Ch=1 TO 24
290       IF Pol_ch(Ch)<>0 THEN Pol_factor(Pol_ch(Ch))=Pol_factor(Pol_ch(Ch))*(-
1)
300     NEXT Ch
310   END IF
320   !
330   Pointy=9
340   LOAD "AUTOST"
350   !
360   END
```

```
10    !  AMPLITUDE : Spherical correction and trace balancing routines
20    !
30    !  January 1987   ---   U.P.   ---  A.T.Dippenaar
40    !
50    !  Last changes 13 January 1988
60    !=============================================================================
70    !
80    OPTION BASE 1
90    !
100   COM /Seis_int/ Line$[2],INTEGER Trace(24,960),Opt,Amt,Gain(24)
110   COM /Seis_real/ REAL Shot(24),Geo(24),Shift_value(24),Offset
120   COM /Processed/ REAL Processed_trace(24,959),Pointy,V
130   COM /Polar/ INTEGER Pol_factor(24)
140   COM /Statics/ REAL Geophone_static(100),Source_static(100),Peg(100)
150   COM /Screens/ INTEGER Screen_1(7500),Screen_2(7500),REAL X,Y,Gain_amplitud
e,Gain_time
160   !
170   !=============================================================================
180   REAL Amplitude_sum
190   DIM Trace_mean(24),Start_spherical(24),Start_ramp(24),End_mute(24),T0_time
(24)
200   !
210   X=100*MAX(1,RATIO)
220   Y=100*MAX(1,1/RATIO)
230   !
240   OUTPUT KBD;"K";
250   Sweep=192
260   INPUT "Sweep length  [ DEF : 192 ]",Sweep
270   Sample_interval=Sweep/959
280   !
290   !-----------------------------------------------------------------------------
300   Ques$="N"
310   INPUT "Do you want to mute the first breaks [ DEF = N ]",Ques$
320   IF Ques$="N" THEN
330     FOR Tr=1 TO Amt
340       DISP "Specify T0 ( time in msec where processes will start) for trace
";Tr;
350       INPUT T0_time(Tr)
360     NEXT Tr
370   END IF
380   !
390   Qqqq$="Y"
400   INPUT "Apply spherical spreading correction [ DEF = Y ]",Qqqq$
410   IF Qqqq$="Y" THEN
420     Sph_type$="E"
430     INPUT "Linear (L) or exponential (E) spreading correction [ DEF : E ]",S
ph_type$
440     IF Sph_type$="E" THEN
450       Kk=5
460       INPUT "Value for 'K' in A = AK(t^n)e^(aT)  [ DEF : 5 ]",Kk
470       Nn=1
480       INPUT "Value for 'n' in A = AK(t^n)e^(aT)  [ DEF : 1 ]",Nn
490       Aa=.5
500       INPUT "Value for 'a' in A = AK(t^n)e^(aT)  [ DEF : 0.5 ]",Aa
510     ELSE
520       IF Sph_type$<>"L" THEN GOTO 430
530     END IF
540   END IF
550   !
560   Qq$="Y"
```

```
570   INPUT "Do you want to normalize the traces [ DEF = Y ]",Qq$
580   !
590   Qqq$="N"
600   INPUT "Balance traces by sliding window  (AGC) [ DEF = N ]",Qqq$
610   T=60
620   IF Qqq$="Y" THEN INPUT "Trace balancing window size (msec) [ DEF = 60 ]",T
630   !=============================================================================
640 Muting:  !
650   IF Ques$="N" THEN
660     PRINT TABXY(25,9);" One moment please .......  "
670     FOR Tr=1 TO Amt
680       End_mute(Tr)=T0_time(Tr)
690       FOR I=1 TO 959
700         Processed_trace(Tr,I)=Trace(Tr,I)
710       NEXT I
720     NEXT Tr
730   END IF
740   !
750   IF Ques$="Y" THEN
760     GRAPHICS ON
770     GLOAD Screen_2(*)
780     VIEWPORT 1/30*X,27/30*X,28,100
790     WINDOW 0,100,-959*Gain_time,50*Gain_time
800     LINE TYPE 4
810     FOR Tim=10 TO 90 STEP 10
820       Position=Tim*959/Sweep
830       MOVE 6,-Position
840       DRAW 100,-Position
850     NEXT Tim
860     LINE TYPE 1
870     !
880     INPUT "Length of muting ramp (msec)",Mute_ramp       ! In time
890     Ramp_length=INT(Mute_ramp/Sample_interval)           ! Sample amount
900     FOR Tr=1 TO Amt
910       DISP "Enter start of muting ramp for trace (msec)";Tr;
920       INPUT Mute
930       T0_time(Tr)=Mute
940       Start_ramp(Tr)=INT(Mute/Sample_interval)
950       End_mute(Tr)=Start_ramp(Tr)+Ramp_length
960     NEXT Tr
970     !
980     GCLEAR
990     FOR Tr=1 TO Amt
1000      PRINT TABXY(25,9);" Muting trace ";Tr
1010      FOR I=1 TO Start_ramp(Tr)
1020        Processed_trace(Tr,I)=0
1030      NEXT I
1040      !
1050      FOR I=Start_ramp(Tr) TO End_mute(Tr)
1060        Ramp_factor=(I-Start_ramp(Tr))/Ramp_length
1070        Processed_trace(Tr,I)=Ramp_factor*Trace(Tr,I)
1080      NEXT I
1090      !
1100      FOR I=End_mute(Tr) TO 959
1110        Processed_trace(Tr,I)=Trace(Tr,I)
1120      NEXT I
1130    NEXT Tr
1140     !
1150  END IF
1160  !=============================================================================
```

```
1170  IF Qqqq$="Y" THEN
1180 Spherical: !
1190    IF Sph_type$="L" THEN
1200      FOR Tr=1 TO Amt
1210        T0=End_mute(Tr)
1220        PRINT TABXY(12,9);" Applying spherical correction -- Trace ";Tr
1230        FOR I=T0 TO 959
1240          Spher_cor=I/T0*Processed_trace(Tr,I)
1250          Processed_trace(Tr,I)=Spher_cor
1260        NEXT I
1270      NEXT Tr
1280    END IF
1290    !
1300    IF Sph_type$="E" THEN
1310      FOR Tr=1 TO Amt
1320        Ref_time=End_mute(Tr)*Sample_interval
1330        PRINT TABXY(12,9);" Applying spherical correction -- Trace ";Tr
1340        FOR I=1 TO 959
1350          Time_val=I*Sample_interval/1000
1360          Tau_val=Time_val/T0_time(Tr)
1370          Processed_trace(Tr,I)=Processed_trace(Tr,I)*Kk*(Time_val^Nn)*EXP(A
a*Tau_val)
1380        NEXT I
1390      NEXT Tr
1400    END IF
1410  END IF
1420  !=============================================================================
1430 Normalize: !
1440  OUTPUT KBD;"K";
1450  IF Qq$="N" THEN 1680
1460  !
1470  FOR Tr=1 TO Amt
1480    PRINT TABXY(22,9);" Normalizing trace ";Tr
1490    Start=End_mute(Tr)
1500    Amplitude_sum=0
1510    FOR J=Start TO 959
1520      Amplitude_sum=ABS(Processed_trace(Tr,J))+Amplitude_sum
1530    NEXT J
1540    Trace_mean(Tr)=Amplitude_sum/(959-Start)
1550  NEXT Tr
1560  !
1570  Ref_amplitude=8000
1580 !Ref_amplitude=MAX(Trace_mean(*))
1590  FOR Tr=1 TO Amt
1600    Trace_scaler=Ref_amplitude/Trace_mean(Tr)
1610    PRINT Trace_scaler
1620    FOR I=Start TO 959
1630      Processed_trace(Tr,I)=Processed_trace(Tr,I)*Trace_scaler
1640    NEXT I
1650  NEXT Tr
1660  !=============================================================================
1670 Balance: !
1680  OUTPUT KBD;"K";
1690  IF Qqq$="N" THEN 2070
1700  Ref_amplitude=1000
1710  Window_length=INT(T/Sample_interval)
1720  !
1730  FOR Tr=1 TO Amt
1740    PRINT TABXY(23,9);" Balancing trace ";Tr
1750    Start=End_mute(Tr)
```

```
1760    Amplitude_sum=0
1770    FOR J=Start TO Start+Window_length-1
1780      IF Processed_trace(Tr,J)=-32768 THEN Processed_trace(Tr,J)=-32767
1790      Amplitude_sum=ABS(Processed_trace(Tr,J))+Amplitude_sum
1800    NEXT J
1810    First_mean=Amplitude_sum/Window_length
1820    First_scaler=Ref_amplitude/First_mean
1830    FOR I=Start TO Window_length/2+Start-1
1840      Processed_trace(Tr,I)=Processed_trace(Tr,I)
1850    NEXT I
1860      !
1870    FOR Move_up=Start TO 959-Window_length-1
1880      Amplitude_sum=0
1890      FOR J=Move_up TO Window_length+Move_up
1900        Amplitude_sum=ABS(Processed_trace(Tr,J))+Amplitude_sum
1910      NEXT J
1920      Mean=Amplitude_sum/Window_length
1930      Scaler=Ref_amplitude/Mean/First_scaler
1940      Index=Move_up+INT(Window_length/2)
1950      Processed_trace(Tr,Index)=Processed_trace(Tr,Index)*Scaler
1960    NEXT Move_up
1970      !
1980    Taper=0
1990    FOR J=959-Window_length/2 TO 959
2000      Processed_trace(Tr,J)=Processed_trace(Tr,J)*Scaler*(Window_length-Tape
r)/Window_length
2010      Taper=Taper+1
2020    NEXT J
2030  NEXT Tr
2040  !
2050  !=============================================================================
2060  !
2070  Pointy=9
2080  FOR Tr=1 TO Amt
2090    FOR I=1 TO 959
2100      IF Processed_trace(Tr,I)>32766 THEN Processed_trace(Tr,I)=32766
2110      IF Processed_trace(Tr,I)<-32765 THEN Processed_trace(Tr,I)=-32765
2120      Trace(Tr,I)=INT(Processed_trace(Tr,I))
2130    NEXT I
2140  NEXT Tr
2150  LOAD "AUTOST:,700"
2160  !
2170  END
```

```
10    !   VEL_ANAL  --  Velocity analysis program
20    !
30    !   A.T.Dippenaar   ---   U.P.  ---   April 1987
40    !
50    !   Last changes on 11 January 1988
60    !=============================================================================
70    !
80    OPTION BASE 1
90    !
100   COM /Seis_int/ Line$[2],INTEGER Trace(24,960),Opt,Amt,Gain(24)
110   COM /Seis_real/ REAL Shot(24),Geo(24),Shift_value(24),Offset
120   COM /Processed/ REAL Processed_trace(24,959),Pointy,V
130   COM /Polar/ INTEGER Pol_factor(24)
140   COM /Statics/ REAL Geophone_static(100),Source_static(100),Peg(100)
150   COM /Screens/ INTEGER Screen_1(7500),Screen_2(7500),REAL X,Y,Gain_amplitud
e,Gain_time
160   !
170   !=============================================================================
180   INTEGER Dummy(961),Sweep,Final_cvs(959),Cvg(959),Mute(25)
190   DIM T(959),Cvs(959),Marker$[6]
200   !
210   !*****************************************************************************
220   !
230   CONTROL 1,12;1
240   Marker$="==>"&CHR$(8)&CHR$(8)&CHR$(8)
250   Pointy=5
260   OUTPUT KBD;"K";
270   Sweep=192
280   !
290   INPUT "Two character line identification",Line$
300   Stat_ques$="Y"
310   INPUT "Must static correction be applied [ DEF = Y ]",Stat_ques$
320   !
330   IF Stat_ques$="Y" THEN
340     !
350     DISP "First peg position on line ";Line$;
360     INPUT First_peg
370     DISP "Last peg position on line ";Line$;
380     INPUT Last_peg
390     Peg_space=5
400     INPUT "Peg spacing [ DEF = 5 ]",Peg_space
410     Peg_amt=(Last_peg-First_peg)/Peg_space+1
420     FOR P=1 TO Peg_amt
430       Peg(P)=First_peg+(P-1)*Peg_space
440     NEXT P
450     !
460     MASS STORAGE IS ":,700,1"
470     Fil$=Line$&"_srce_st"
480     ASSIGN @P TO Fil$
490     ON END @P GOTO 510
500     ENTER @P;Source_static(*)
510     ASSIGN @P TO *
520     !
530     Fil$=Line$&"_geo_st"
540     ASSIGN @P TO Fil$
550     ON END @P GOTO 570
560     ENTER @P;Geophone_static(*)
570     ASSIGN @P TO *
580     !
590   END IF
```

```
600   !
610   PRINT TABXY(27,2);CHR$(129)&" Velocity analysis options "&CHR$(128)
620   PRINT TABXY(22,5);" 1 ..... CVG  --  constant velocity gathers"
630   PRINT TABXY(22,7);" 2 ..... CVS  --  constant velocity stack"
640   PRINT TABXY(22,9);" 3 ..... RETURN to main menu"
650   PRINT TABXY(19,Pointy);Marker$;
660   !
670   ON KNOB .15 GOSUB Move_marker
680   ON KBD GOSUB Keyboard
690 Spin:GOTO Spin
700   !
710 Move_marker:  !
720   PRINT TABXY(19,Pointy);"
730   IF KNOBX>0 THEN
740     Pointy=Pointy+2
750     IF Pointy>9 THEN Pointy=9
760   ELSE
770     Pointy=Pointy-2
780     IF Pointy<5 THEN Pointy=5
790   END IF
800   PRINT TABXY(19,Pointy);Marker$;
810   RETURN
820   !
830 Keyboard:  !
840   PRINT TABXY(19,Pointy);"
850   SELECT KBD$
860   CASE CHR$(255)&"^"
870     Pointy=Pointy-2
880     IF Pointy<5 THEN Pointy=9
890   CASE CHR$(255)&"V"
900     Pointy=Pointy+2
910     IF Pointy>9 THEN Pointy=5
920   CASE CHR$(255)&"X",CHR$(255)&"E",CHR$(255)&"C"
930     SELECT Pointy
940     CASE 5
950       Anal_chce$="CVG"
960     CASE 7
970       Anal_chce$="CVS"
980     CASE 9
990       LOAD "AUTOST:,700,0  "
1000    END SELECT
1010    OUTPUT KBD;"K";
1020    GOTO 1090
1030  CASE ELSE
1040  END SELECT
1050  PRINT TABXY(19,Pointy);Marker$;
1060  RETURN
1070  !
1080  !*************************************************************************
1090  Sample_interval=192/959              ! Sample interval in msec
1100  Si=Sample_interval/1000              ! Sample interval in sec
1110  !
1120  Cdp_fold=6
1130  INPUT "Fold of proposed CDP stack  [  DEF = 6 ]",Cdp_fold
1140  Amt=Cdp_fold
1150  Shot_space=5
1160  INPUT "Shot spacing [ DEF = 5 ]",Shot_space
1170  !
1180  Ques$="Y"
1190  Nr_midpoints=1
```

```
1200  !
1210  IF Anal_chce$="CVS" THEN
1220    Nr_midpoints=5
1230    INPUT "How many midpoints must be stacked with every velocity [  Def = 5
  ]",Nr_midpoints
1240    Zero_trace_ques$="N"
1250    INPUT "Must the zero offset trace be included in the stack [ DEF = N ]",
Zero_trace_ques$
1260    INPUT "Shotpoint location of first CMP",First_cdp
1270    Ques$="N"
1280    INPUT "Do you want to store the corrected pre-stack traces   [ DEF = N ]
",Ques$
1290    FOR Tr=1 TO Amt
1300      DISP "Enter pre-stack mute (msec) for trace ";Tr;
1310      INPUT Mute_time
1320      Mute(Tr)=INT(Mute_time*959/Sweep)
1330    NEXT Tr
1340    Mute(Amt+1)=959
1350  END IF
1360  !
1370  IF Anal_chce$="CVG" THEN INPUT "Shotpoint location of CMP ",First_cdp
1380  !
1390  First_vel=1000
1400  Last_vel=4000
1410  Vel_increment=500
1420  PRINT TABXY(1,18);"Enter stacking velocity range in m/s (First,Last,Increm
ent) "
1430  INPUT "[ DEF: 1000,4000,200 ]",First_vel,Last_vel,Vel_increment
1440  OUTPUT KBD;"K";
1450  !
1460  FOR Tr=1 TO Amt
1470    Centre=First_cdp+(Tr-1)*Shot_space
1480    Shot(Tr)=Centre-((Tr-1)*Shot_space)
1490    Geo(Tr)=Centre+((Tr-1)*Shot_space)
1500  NEXT Tr
1510  !
1520  !=============================================================================
1530  !  Routine to read in previously calculated NMO corrections for given
1540  !  velocities and offsets
1550  !=============================================================================
1560  MASS STORAGE IS ":,700,1  "
1570  CAT TO #CRT;NO HEADER,SELECT "v",COUNT Count
1580  IF Count=0 THEN 1640
1590  ALLOCATE Prev_nmo$(1:Count)[80]
1600  OUTPUT KBD;"K";
1610  PRINT TABXY(15,9);"Reading in previous calculated filenames"
1620  CAT TO Prev_nmo$(*);NO HEADER,SELECT "v"
1630  !
1640  FOR Velocity=First_vel TO Last_vel STEP Vel_increment
1650    !
1660    V=Velocity/1000                          ! velocity in m/msec
1670    !
1680    FOR Tr=2 TO Amt
1690      Seperation=ABS(Geo(Tr)-Shot(Tr))
1700      Filename$="v"&VAL$(Velocity)&"o"&VAL$(Seperation)
1710      FOR C=1 TO Count
1720        Length=POS(Prev_nmo$(C)," ")-1
1730        IF Filename$=Prev_nmo$(C)[1,Length] THEN 1930
1740      NEXT C
1750      !=============================================================================
```

```
1760      !  Generate new T^2 = T0^2 + ( X^2 - V^2 ) values and store on disc
1770      !=================================================================
1780      PRINT TABXY(20,9);"Generating new NMO values for offset ";Seperation;"
          "
1790      Xv=Seperation^2/V^2
1800      FOR I=1 TO 959
1810        T0=(I*Sample_interval)^2
1820        T(I)=SQR(T0+Xv)
1830      NEXT I
1840      !
1850      OUTPUT KBD;"K";
1860      PRINT TABXY(32,9);"Storing ";Filename$;" on disc"
1870      MASS STORAGE IS ":,700,1"
1880      CREATE BDAT Filename$,4,1918
1890      ASSIGN @P TO Filename$
1900      OUTPUT @P;T(*)
1910      ASSIGN @P TO *
1920      !
1930    NEXT Tr
1940  NEXT Velocity
1950  !
1960  MASS STORAGE IS ":,700,1  "
1970  FOR Cdp=1 TO Nr_midpoints
1980    !
1990    Cdp_centre=First_cdp+(Cdp-1)*Shot_space
2000    OUTPUT KBD;"K";
2010    Ms$=":,700,0,2"
2020    MASS STORAGE IS Ms$
2030    FOR Tr=1 TO Amt
2040      Shot(Tr)=Cdp_centre-((Tr-1)*Shot_space)
2050      Geo(Tr)=Cdp_centre+((Tr-1)*Shot_space)
2060      GOSUB Read_trace
2070      !
2080      IF Stat_ques$="Y" THEN
2090        PRINT TABXY(22,9);"Doing static correction for trace ";Tr
2100        Total_static(Tr)=0
2110        FOR P=1 TO Peg_amt
2120          IF Shot(Tr)=Peg(P) THEN Total_static(Tr)=Total_static(Tr)+Source_s
tatic(P)
2130          IF Geo(Tr)=Peg(P) THEN Total_static(Tr)=Total_static(Tr)+Geophone_
static(P)
2140        NEXT P
2150        Sample_shift(Tr)=INT(Total_static(Tr)/Si)
2160        !-----------------------------------------------------------------
2170        !            Applying static corrections
2180        !-----------------------------------------------------------------
2190        FOR I=1 TO 959-Sample_shift(Tr)
2200          Trace(Tr,I)=Trace(Tr,I+Sample_shift(Tr))
2210        NEXT I
2220        !-----------------------------------------------------------------
2230        FOR I=959-Sample_shift(Tr) TO 959
2240          Trace(Tr,I)=0
2250        NEXT I
2260      END IF
2270    NEXT Tr
2280    !
2290    FOR V=First_vel TO Last_vel STEP Vel_increment
2300      !
2310      FOR Tr=1 TO Amt
2320        IF Shot(Tr)=Geo(Tr) THEN
```

```
2330          Position=Shot(Tr)
2340          FOR I=1 TO 959
2350            IF Zero_trace_ques$="Y" THEN
2360              Cvs(I)=Trace(Tr,I)
2370            ELSE
2380              Cvs(I)=0
2390            END IF
2400            Cvg(I)=Trace(Tr,I)
2410          NEXT I
2420          GOTO 2880
2430        END IF
2440        !
2450        !=====================================================================
2460        !    Read in calculated values of NMO corrections
2470        !=====================================================================
2480        OUTPUT KBD;"K";
2490        Seperation=ABS(Geo(Tr)-Shot(Tr))
2500        Filename$="v"&VAL$(V)&"o"&VAL$(Seperation)
2510        DISP "Reading in ";Filename$
2520        !
2530        MASS STORAGE IS ":,700,1  "
2540        ASSIGN @In TO Filename$
2550        ENTER @In;T(*)
2560        ASSIGN @In TO *
2570        !=====================================================================
2580        !   Shifting values to NMO corrected sample positions.
2590        !=====================================================================
2600        Sample_inverse=1/Sample_interval
2610        OUTPUT KBD;"K";
2620        PRINT TABXY(30,9);"Doing NMO correction ";Tr
2630        FOR I=1 TO 959
2640          Remainder=FRACT(T(I)/Sample_interval)*Sample_interval
2650          T1=T(I)-Remainder
2660          T2=T1+Sample_interval
2670          Index1=T1*Sample_inverse
2680          Index2=T2*Sample_inverse
2690          !===================================================================
2700          !     T1,T2 are TWT's (multiples of sample interval) which
2710          !     straddles calculated NMO correction. Index1 and Index2
2720          !     are corresponding sample numbers
2730          !-------------------------------------------------------------------
2740          IF Index1>958 THEN 2860
2750          Difference=(Trace(Tr,Index2)-Trace(Tr,Index1))*(Remainder*Sample_i
nverse)
2760          !-------------------------------------------------------------------
2770          !     Difference is interpolated value between T1 and T2
2780          !===================================================================
2790          Cvg(I)=INT(Trace(Tr,Index1)+Difference)
2800          !
2810          IF Anal_chce$="CVS" THEN
2820            IF I<Mute(Tr) THEN 2860
2830            Cvs(I)=Cvs(I)+Trace(Tr,Index1)+Difference
2840          END IF
2850          !
2860        NEXT I
2870        !
2880        IF Ques$="Y" THEN
2890          MASS STORAGE IS ":,700,0  "
2900          Fil_cvg$=VAL$(Shot(Tr))&"v"&VAL$(V/100)&"g"&VAL$(Geo(Tr))
2910          CREATE BDAT Fil_cvg$,1,1950
```

```
2920          ASSIGN @Out TO Fil_cvg$
2930          OUTPUT @Out;Cvg(*)
2940          ASSIGN @Out TO *
2950          MASS STORAGE IS ":,700,1"
2960        END IF
2970        !
2980      NEXT Tr
2990      !
3000      IF Anal_chce$="CVG" THEN 3290
3010      !
3020      !-------------------------------------------------------
3030      !   CVS stack fold compensation
3040      !-------------------------------------------------------
3050      Tr=1
3060      IF Zero_trace_ques$="N" THEN Tr=2
3070      FOR I=1 TO 959
3080        IF I>=Mute(Tr) THEN Tr=Tr+1
3090        IF Zero_trace_ques$="N" THEN
3100          Cmp_temp=INT(Cvs(I)/SQR(Tr-1))
3110        ELSE
3120          Cmp_temp=INT(Cvs(I)/SQR(Tr))
3130        END IF
3140        IF ABS(Cmp_temp)>32767 THEN
3150          IF Cmp_temp>32767 THEN Final_cvs(I)=32767
3160          IF Cmp_temp<-32766 THEN Final_cvs(I)=-32766
3170        ELSE
3180          Final_cvs(I)=Cmp_temp
3190        END IF
3200      NEXT I
3210      !
3220      MASS STORAGE IS ":,700,0  "
3230      Filename$=VAL$(V)&"cvs"&VAL$(Position)
3240      CREATE BDAT Filename$,1,1950
3250      ASSIGN @Out TO Filename$
3260      OUTPUT @Out;Final_cvs(*)
3270      ASSIGN @Out TO *
3280      !
3290    NEXT V
3300  NEXT Cdp
3310  !
3320  MASS STORAGE IS ":,700,1  "
3330  LOAD "AUTOST:,700"
3340  !
3350  STOP
3360!=============================================================================
3370 Read_trace:!
3380  !
3390  MASS STORAGE IS ":,700,0"
3400  IF Shot(Tr)<0 THEN
3410    Fil$=Line$&"n"&VAL$(ABS(Shot(Tr)))&"_"&VAL$(Geo(Tr))
3420  ELSE
3430    Fil$=Line$&"_"&VAL$(Shot(Tr))&"_"&VAL$(Geo(Tr))
3440  END IF
3450  DISP "Reading trace ";Tr;" of ";Amt;"     ----    ";Fil$
3460  !
3470  ON ERROR GOSUB To_err_is_human
3480  ASSIGN @To_disc TO Fil$
3490  ON END @To_disc GOTO 3510
3500  ENTER @To_disc;Dummy(*)
3510  FOR I=1 TO 959
```

```
3520     Trace(Tr,I)=Dummy(I)
3530   NEXT I
3540   Gain(Tr)=Dummy(960)
3550   ASSIGN @To_disc TO *
3560   MASS STORAGE IS ":,700,1"
3570   OFF ERROR
3580   !
3590   RETURN
3600   !-------------------------------------------------------------------------
3610   !
3620 To_err_is_human: !
3630   SELECT ERRN
3640   CASE 56
3650     Ms_counter=Ms_counter+1
3660     !
3670     SELECT Ms$
3680     CASE ":,700,0,2"
3690       Ms$=":,700,0,3"
3700     CASE ":,700,0,3"
3710       Ms$=":,700,1"
3720     CASE ":,700,1"
3730       Ms$=":,700,0,2"
3740     END SELECT
3750 !   MASS STORAGE IS Ms$
3760     !
3770     IF Ms_counter>4 THEN
3780       BEEP
3790       Ms_counter=0
3800       PRINT TABXY(20,14);" Couldn't find ";Fil$;" on current MSU"
3810       INPUT "do you want to change the MSU",Msu_ques$
3820       SELECT Msu_ques$
3830       CASE "Y"
3840         INPUT "Enter new MSU  - e.g. '':,700,0,3'' [ NO DEF ]",Ms$
3850         ON ERROR GOTO 3890
3860         MASS STORAGE IS Ms$
3870         OFF ERROR
3880         GOTO 3920
3890         BEEP
3900         DISP "TRY AGAIN Bimbo ----";
3910         GOTO 3840
3920       CASE "N"
3930         DISP "Insert next disc in current MSU   <CONTINUE> "
3940         PAUSE
3950       CASE ELSE
3960         BEEP
3970         GOTO 3810
3980       END SELECT
3990     END IF
4000   END SELECT
4010   RETURN
4020   !=========================================================================
4030   END
```

```
10     !   NMO_CORR_1  --  Normal moveout corrections  --  plotting routines
20     !
30     !   A.T.Dippenaar   ---   U.P.  ---   April 1987
40     !
50     !   Last changes on 10 January 1988
60     !=============================================================================
70     !
80     OPTION BASE 1
90     !
100    COM /Seis_int/ Line$[2],INTEGER Trace(24,960),Opt,Amt,Gain(24)
110    COM /Seis_real/ REAL Shot(24),Geo(24),Shift_value(24),Offset
120    COM /Processed/ REAL Processed_trace(24,959),Pointy,V
130    COM /Polar/ INTEGER Pol_factor(24)
140    COM /Statics/ REAL Geophone_static(100),Source_static(100),Peg(100)
150    COM /Screens/ INTEGER Screen_1(7500),Screen_2(7500),REAL X,Y,Gain_amplitud
e,Gain_time
160    !
170    !=============================================================================
180    INTEGER Dummy(961),Mute(25),Store_trace(959),Cmp_store(959),Sample_shift(4
5)
190    DIM Twt(8),Input_vel(8),Stack_vel(959),Nmo_cor_trace(24,959),Cmp_stacked(9
59),Nmo(24,959)
200    DIM Marker$[6]
210    REAL Difference,Diff
220    Marker$="==>"&CHR$(8)&CHR$(8)&CHR$(8)
230    !
240    CONTROL 1,12;1
250    OUTPUT KBD;"K";
260    Sweep=192
270    Sample_interval=Sweep/959                ! Sample interval in msec
280    Si=Sample_interval/1000                  ! Sample interval in sec
290    !
300    Ques$="N"
310    INPUT "Do you want to store the NMO corrected traces (unstacked) [ Def = N
 ]",Ques$
320    DISP "Ensure disc for output files in left hand drive ( '':,700,1'' )   <C
ONTINUE> "
330    PAUSE
340    INPUT "Line identification (Maximum 2 alphanumericals)",Line$
350    !
360    Cdp_fold=6
370    INPUT "Fold of proposed CDP stack [ DEF = 6 ]",Cdp_fold
380    Zero_trace_ques$="N"
390    INPUT "Must the zero offset trace be included in the stack [ DEF = N ]",Ze
ro_trace_ques$
400    Amt=Cdp_fold
410    Shot_space=5
420    INPUT "Shot spacing  [ DEF = 5 ]",Shot_space
430    INPUT "First CDP midpoint (shotpoint location)",First_cdp
440    Position=First_cdp
450    !
460    Stat_ques$="Y"
470    INPUT "Must static correction be applied [ DEF = Y ]",Stat_ques$
480    !
490    IF Stat_ques$="Y" THEN
500      !
510      INPUT "First peg position",First_peg
520      INPUT "Last peg position",Last_peg
530      Peg_space=5
540      INPUT "Peg spacing [ DEF = 5 ]",Peg_space
```

```
550     Peg_amt=(Last_peg-First_peg)/Peg_space+1
560     FOR P=1 TO Peg_amt
570       Peg(P)=First_peg+(P-1)*Peg_space
580     NEXT P
590     !
600     MASS STORAGE IS ":,700,1  "
610     !
620     Fil$=Line$&"_srce_st"
630     ASSIGN @P TO Fil$
640     ON END @P GOTO 660
650     ENTER @P;Source_static(*)
660     ASSIGN @P TO *
670     !
680     Fil$=Line$&"_geo_st"
690     ASSIGN @P TO Fil$
700     ON END @P GOTO 720
710     ENTER @P;Geophone_static(*)
720     ASSIGN @P TO *
730     !
740   END IF
750   !
760   FOR Tr=1 TO Amt
770     Shot(Tr)=First_cdp-((Tr-1)*Shot_space)
780     Geo(Tr)=First_cdp+((Tr-1)*Shot_space)
790   NEXT Tr
800   !=============================================================================
810   PRINT TABXY(14,14);"  First TWT-velocity pair must be at time = 0 msec"
820   PRINT TABXY(14,16);"  Last TWT-velocity pair must be at time = 192 msec"
830   !
840   INPUT "How many TWT -  Velocity pairs are you entering",Nr_pairs
850   IF Nr_pairs>8 THEN
860     BEEP
870     DISP "Let's be reasonable   --  Try for less than 8 please"
880     BEEP
890     WAIT 1.5
900     GOTO 840
910   END IF
920   !
930   DISP "TWT,velocity pair # 1";
940   INPUT Twt(1),Input_vel(1)
950   IF Twt(1)<>0 THEN 900
960   !
970   OUTPUT KBD;"K";
980   PRINT TABXY(20,5);" TWT          !    Stacking velocity"
990   PRINT TABXY(20,6);"-------------------------------------"
1000  PRINT TABXY(33,7);"!"
1010  PRINT TABXY(21,7);Twt(1)
1020  PRINT TABXY(40,7);Input_vel(1)
1030  !
1040  FOR I=2 TO Nr_pairs
1050    DISP "TWT,velocity pair # ";I;
1060    INPUT Twt(I),Input_vel(I)
1070    PRINT TABXY(33,6+I);"!"
1080    PRINT TABXY(21,6+I);Twt(I)
1090    PRINT TABXY(40,6+I);Input_vel(I)
1100  NEXT I
1110  !
1120  IF Twt(I-1)<>192 THEN
1130    BEEP
1140    DISP "Last TWT is  not equal to 192 msec  --  Redo TWT,VELOCITY input"
```

```
1150    BEEP
1160    WAIT 1.5
1170    OUTPUT KBD;"K";
1180    GOTO 840
1190  END IF
1200  OUTPUT KBD;"K";
1210  !=============================================================================
1220  !
1230  !   Calculating stacking velocity for every sample point.  Interpolation
1240  !   to fill velocity values between input velocities.
1250  !
1260  !-----------------------------------------------------------------------------
1270  PRINT TABXY(23,9);"Interpolating stacking velocities"
1280  FOR V=1 TO Nr_pairs-1
1290    Diff=Sample_interval*(Input_vel(V+1)-Input_vel(V))/(Twt(V+1)-Twt(V))
1300    Index1=Twt(V)/Sample_interval
1310    IF Index1=0 THEN Index1=1
1320    Index2=Twt(V+1)/Sample_interval
1330    Stack_vel(Index1)=Input_vel(V)                          !   in m/s
1340    FOR Vv=Index1+1 TO Index2
1350      Stack_vel(Vv)=Stack_vel(Vv-1)+Diff
1360    NEXT Vv
1370  NEXT V
1380  Stack_vel(959)=Input_vel(Nr_pairs)
1390  !=============================================================================
1400  !
1410  OUTPUT KBD;"K";
1420  PRINT TABXY(28,9);"Gathering CMP traces"
1430  MASS STORAGE IS ":,700,1  "
1440  FOR Tr=1 TO Amt
1450    ON ERROR GOSUB To_err_is_human
1460    GOSUB Read_trace
1470    OFF ERROR
1480  NEXT Tr
1490  Max_offset=ABS(Shot(Amt)-Geo(Amt))
1500  !
1510  !=============================================================================
1520  IF Stat_ques$="Y" THEN
1530    FOR Tr=1 TO Amt
1540      PRINT TABXY(22,9);"Applying static corrections  --  trace ";Tr
1550      Total_static(Tr)=0
1560      FOR P=1 TO Peg_amt
1570        IF Shot(Tr)=Peg(P) THEN Total_static(Tr)=Total_static(Tr)+Source_sta
tic(P)
1580        IF Geo(Tr)=Peg(P) THEN Total_static(Tr)=Total_static(Tr)+Geophone_st
atic(P)
1590      NEXT P
1600      Sample_shift(Tr)=INT(Total_static(Tr)/Si)
1610      !
1620      !             Applying static corrections
1630      !
1640      FOR I=1 TO 959-Sample_shift(Tr)
1650        Trace(Tr,I)=Trace(Tr,I+Sample_shift(Tr))
1660      NEXT I
1670      !
1680      FOR I=959-Sample_shift(Tr) TO 959
1690        Trace(Tr,I)=0
1700      NEXT I
1710      !
1720    NEXT Tr
```

```
1730  END IF
1740  !
1750  !=============================================================================
1760  FOR Tr=1 TO Amt
1770    PRINT TABXY(20,9);"Generating NMO values :-- offset ";Tr;"
  "
1780    !
1790    Seperation=ABS(Geo(Tr)-Shot(Tr))
1800    FOR I=1 TO 959
1810      T0=(I*Sample_interval)^2                ! in msec
1820      V2=(Stack_vel(I)/1000)^2                !  m/msec
1830      Nmo(Tr,I)=SQR(T0+Seperation^2/V2)       ! in msec
1840    NEXT I
1850    !
1860  NEXT Tr
1870  !
1880  !=============================================================================
1890  !   Shifting values to NMO corrected sample positions.
1900  !=============================================================================
1910  OUTPUT KBD;"K";
1920  FOR Tr=1 TO Amt
1930    PRINT TABXY(26,9);"Doing NMO correction ";Tr
1940    FOR I=1 TO 959
1950      Remainder=FRACT(Nmo(Tr,I)/Sample_interval)*Sample_interval
1960      T1=Nmo(Tr,I)-Remainder
1970      T2=T1+Sample_interval
1980      Index1=T1/Sample_interval
1990      Index2=T2/Sample_interval
2000      !=============================================================================
2010      !     T1,T2 are TWT's (multiples of sample interval) which
2020      !     straddles calculated NMO correction. Index1 and Index2
2030      !     are corresponding sample numbers
2040      !
2050      !     'Difference' is interpolated value between T1 and T2
2060      !-----------------------------------------------------------------------
2070      IF Index1>958 THEN 2220
2080      ON ERROR GOTO 2110
2090      Diff=Trace(Tr,Index2)-Trace(Tr,Index1)
2100      GOTO 2190
2110      IF ERRN=20 THEN
2120        OFF ERROR
2130        Diff=0
2140        GOTO 2190
2150      ELSE
2160        DISP "ooops ----  ERRN =";ERRN
2170        PAUSE
2180      END IF
2190      Difference=Diff*(Remainder/Sample_interval)
2200      Nmo_cor_trace(Tr,I)=Trace(Tr,Index1)+Difference
2210      Store_trace(I)=INT(Trace(Tr,Index1)+Difference)
2220    NEXT I
2230    PRINT TABXY(15,9);"
2240    !-----------------------------------------------------------------------
2250    !   Storing NMO corrected traces (unstacked) on disc if required
2260    !=============================================================================
2270    IF Ques$="Y" THEN
2280      MASS STORAGE IS ":,700,1   "
2290      Fil$=VAL$(Shot(Tr))&"nmo"&VAL$(Geo(Tr))
2300      PRINT TABXY(16,9);"Storing NMO corrected trace ";Tr;" in ";Fil$
2310      CREATE BDAT Fil$,1,1950
```

```
2320      ASSIGN @To_disc TO Fil$
2330      ON END @To_disc GOTO 2350
2340      OUTPUT @To_disc;Store_trace(*)
2350      ASSIGN @To_disc TO *
2360      OUTPUT KBD;"K";
2370    END IF
2380    !
2390  NEXT Tr
2400  !=============================================================================
2410  !   Plotting and stacking routines
2420  !=============================================================================
2430  GINIT
2440  GRAPHICS ON
2450  X=100*MAX(1,RATIO)
2460  Y=100*MAX(1,1/RATIO)
2470  !
2480  OFF ERROR
2490  GOSUB Menu
2500 Spinner:GOTO Spinner
2510  !
2520  STOP
2530  !=============================================================================
2540 Menu: !
2550  Pointy=7
2560  GCLEAR
2570  ON KBD GOSUB Keyboard
2580  ON KNOB .25 GOSUB Move_pointer
2590  !
2600  CONTROL 1,12;1
2610  OUTPUT KBD;"K";
2620  PRINT TABXY(32,4);CHR$(129)&" Plotting options "&CHR$(128)
2630  PRINT TABXY(20,7);"1  .....   Plot NMO corrected traces and specify mute"
2640  PRINT TABXY(20,9);"2  .....   Plot NMO functions (just for the fun of it)"
2650  PRINT TABXY(20,11);"3  .....   Print velocities and time - depth curve"
2660  PRINT TABXY(20,13);"4  .....   RETURN to main menu"
2670  PRINT TABXY(16,7);Marker$;
2680  RETURN
2690  !-----------------------------------------------------------------------------
2700 Mute_and_stack: !
2710  !-----------------------------------------------------------------------------
2720  !  Muting routine
2730  !-----------------------------------------------------------------------------
2740  GCLEAR
2750  OUTPUT KBD;"K";
2760  FOR Tr=1 TO Amt
2770    PRINT TABXY(32,9);"Muting trace ";Tr
2780    FOR M=1 TO Mute(Tr)
2790      Nmo_cor_trace(Tr,M)=0
2800    NEXT M
2810  NEXT Tr
2820  !-----------------------------------------------------------------------------
2830  !  Stacking routine
2840  !-----------------------------------------------------------------------------
2850  PRINT TABXY(31,9);"Stacking trace
2860  Begin=2
2870  IF Zero_trace_ques$="Y" THEN Begin=1
2880  FOR Tr=Begin TO Amt
2890    FOR S=1 TO 959
2900      Cmp_stacked(S)=Cmp_stacked(S)+Nmo_cor_trace(Tr,S)
2910    NEXT S
```

```
2920  NEXT Tr
2930  !-------------------------------------------------------------------------
2940  !   Fold compensation and write to INTEGER format
2950  !-------------------------------------------------------------------------
2960  Tr=1
2970  FOR I=1 TO 959
2980    IF I>=Mute(Tr) THEN Tr=Tr+1
2990    Cmp_temp=INT(Cmp_stacked(I)/SQR(Tr))
3000    IF ABS(Cmp_temp)>32767 THEN
3010      IF Cmp_temp>32767 THEN Cmp_store(I)=32767
3020      IF Cmp_temp<-32766 THEN Cmp_store(I)=-32766
3030    ELSE
3040      Cmp_store(I)=Cmp_temp
3050    END IF
3060  NEXT I
3070  !-------------------------------------------------------------------------
3080  !   Storing CMP stacked trace on disc
3090  !-------------------------------------------------------------------------
3100  MASS STORAGE IS ":,700,1"
3110  Fil$=Line$&"_"&VAL$(Position)
3120  DISP Fil$
3130  CREATE BDAT Fil$,1,1950
3140  ASSIGN @To_disc TO Fil$
3150  ON END @To_disc GOTO 3170
3160  OUTPUT @To_disc;Cmp_store(*)
3170  ASSIGN @To_disc TO *
3180  !
3190  GOSUB Menu
3200  RETURN
3210  !-------------------------------------------------------------------------
3220 Read_trace:!
3230  !
3240  DISP TAB(46);"Reading trace ";Tr;" of ";Amt
3250  IF Shot(Tr)<0 THEN
3260    Fil$=Line$&"n"&VAL$(ABS(Shot(Tr)))&"_"&VAL$(Geo(Tr))
3270  ELSE
3280    Fil$=Line$&"_"&VAL$(Shot(Tr))&"_"&VAL$(Geo(Tr))
3290  END IF
3300  !
3310  ASSIGN @To_disc TO Fil$
3320  ON END @To_disc GOTO 3340
3330  ENTER @To_disc;Dummy(*)
3340  FOR I=1 TO 959
3350    Trace(Tr,I)=Dummy(I)
3360  NEXT I
3370  Gain(Tr)=Dummy(960)
3380  ASSIGN @To_disc TO *
3390  OFF ERROR
3400  !
3410  DISP "
        "
3420  RETURN
3430  !-------------------------------------------------------------------------
3440  !
3450 Keyboard: !
3460  OUTPUT KBD;"K";
3470  SELECT KBD$
3480  CASE CHR$(255)&"C",CHR$(255)&"E",CHR$(255)&"X"
3490    SELECT Pointy
3500    CASE 7
```

```
3510      GOSUB Plot_nmo_traces
3520    CASE 9
3530      GOSUB Plot_functions
3540    CASE 11
3550      GOSUB Velocities
3560    CASE 13
3570      MASS STORAGE IS ":,700,0  "
3580      LOAD "AUTOST"
3590    END SELECT
3600  CASE ELSE
3610    BEEP 1800,.4
3620    GOTO 3640
3630  END SELECT
3640  RETURN
3650  !-------------------------------------------------------------------------
3660 Move_pointer: !
3670  PRINT TABXY(16,Pointy);"    ";
3680  IF KNOBX>0 THEN
3690    Pointy=Pointy+2
3700    IF Pointy>13 THEN Pointy=13
3710  ELSE
3720    Pointy=Pointy-2
3730    IF Pointy<7 THEN Pointy=7
3740  END IF
3750  PRINT TABXY(16,Pointy);Marker$;
3760  RETURN
3770  !-------------------------------------------------------------------------
3780 Plot_nmo_traces: !
3790  LORG 6
3800  CSIZE 4
3810  MOVE .5*X,.98*Y
3820  LABEL " NMO corrected traces  --  Line "&Line$&" : CMP "&VAL$(Position)
3830  !
3840  VIEWPORT .25*X,.98*X,.15*Y,.9*Y
3850  WINDOW 0,(12*64000.),-480,0
3860  FOR Tr=1 TO Amt
3870    MOVE (Tr-1)*64000.+32000,0
3880    LORG 4
3890    CSIZE 2.8
3900    LABEL Tr
3910    FOR I=1 TO 480
3920      IF Nmo_cor_trace(Tr,I)>0 THEN MOVE ((Tr-1)*64000.)+32000,-I
3930      DRAW Nmo_cor_trace(Tr,I)+((Tr-1)*64000.)+32000,-I
3940    NEXT I
3950  NEXT Tr
3960  !
3970  Sweep=192
3980  CSIZE 3.0
3990  FOR Ti=0 TO 75 STEP 5
4000    LINE TYPE 4
4010    MOVE 0,-Ti*959/Sweep
4020    DRAW 12*64000,-Ti*959/Sweep
4030    LINE TYPE 1
4040    MOVE 0,-Ti*959/Sweep
4050    CLIP OFF
4060    LORG 8
4070    LABEL Ti
4080    CLIP ON
4090  NEXT Ti
4100  !
```

```
4110  FOR Tr=1 TO Amt
4120    DISP "Enter stretch mute (in msec) for trace";Tr;
4130    INPUT Mute_time                                    ! in msec
4140    Mute(Tr)=INT(Mute_time/Sample_interval)            !  sample #
4150    PRINT TABXY(1,4+Tr);"Mute ";Tr;" = ";INT(Mute_time);
4160  NEXT Tr
4170  Mute(Amt+1)=959
4180  !
4190  INPUT "Satisfied (Y/N)",Sat$
4200  IF Sat$="Y" THEN
4210    GOSUB Mute_and_stack
4220  ELSE
4230    OUTPUT KBD;"K";
4240    GOTO 4110
4250  END IF
4260  RETURN
4270  !-------------------------------------------------------------------------
4280 Plot_functions: !
4290  FOR K=0 TO 9
4300    ON KEY K LABEL "     " GOTO Spinner
4310  NEXT K
4320  ON KEY 7 LABEL "   RETURN    " GOSUB Menu
4330  CONTROL 1,12;0
4340  !
4350  GINIT
4360  LORG 6
4370  CSIZE 4
4380  MOVE .5*X,.98*Y
4390  LABEL "Normal moveout functions  --  Line "&Line$&" : CMP "&VAL$(Position)
4400  VIEWPORT .45*X,.85*X,.02*Y,.8*Y
4410  FRAME
4420  WINDOW 0,Max_offset,-192,0
4430  LINE TYPE 4
4440  GRID 10,25,0,0
4450  LINE TYPE 1
4460  !
4470  CLIP OFF
4480  CSIZE 3.2
4490  LORG 8
4500  FOR T=25 TO 175 STEP 25
4510    MOVE -1,-T
4520    LABEL T
4530  NEXT T
4540  !
4550  LORG 4
4560  FOR O=10 TO Max_offset STEP 10
4570    MOVE O,5
4580    LABEL O
4590  NEXT O
4600  !
4610  LORG 5
4620  CSIZE 2.5
4630  CLIP ON
4640  FOR V=1 TO Nr_pairs
4650    FOR I=1 TO 959
4660      IF Twt(V)=INT(I*Sample_interval) THEN
4670        MOVE 0,-Nmo(1,I)
4680        FOR Tr=1 TO Amt
4690          Offset=ABS(Shot(Tr)-Geo(Tr))
4700          DRAW Offset,-Nmo(Tr,I)
```

```
4710          MOVE Offset,-Nmo(Tr,I)
4720          LABEL "*"
4730          MOVE Offset,-Nmo(Tr,I)
4740        NEXT Tr
4750        GOTO 4780
4760      END IF
4770    NEXT I
4780  NEXT V
4790  !
4800  VIEWPORT 0,.35*X,.02*Y,.8*Y
4810  WINDOW 0,50,-192,0
4820  CLIP OFF
4830  LORG 7
4840  CSIZE 4
4850  MOVE 10,0
4860  LABEL "T"
4870  MOVE 28,0
4880  LABEL "V"
4890  LORG 3
4900  CSIZE 3.0
4910  MOVE 10,0
4920  LABEL "0"
4930  MOVE 28,0
4940  LABEL "stack"
4950  !
4960  MOVE 52,-192
4970  DRAW 52,0
4980  MOVE 51.2,0
4990  DRAW 51.2,-192
5000  !
5010  LORG 6
5020  CSIZE 3.8
5030  FOR I=1 TO Nr_pairs
5040    MOVE 10,-(I*10)-2
5050    LABEL Twt(I)
5060    MOVE 28,-(I*10)-2
5070    LABEL Input_vel(I)
5080  NEXT I
5090  !
5100  RETURN
5110  !-------------------------------------------------------------------------
5120 To_err_is_human: !
5130  SELECT ERRN
5140  CASE 56
5150    BEEP
5160    PRINT TABXY(20,14);" Couldn't find ";Fil$;" on current MSU"
5170    INPUT "do you want to change the MSU",Msu_ques$
5180    SELECT Msu_ques$
5190    CASE "Y"
5200      INPUT "Enter new MSU  - e.g. '':,700,0,3'' [ NO DEF ]",Ms$
5210      ON ERROR GOTO 5250
5220      MASS STORAGE IS Ms$
5230      OFF ERROR
5240      GOTO 5280
5250      BEEP
5260      DISP "TRY AGAIN Bimbo ----";
5270      GOTO 5200
5280    CASE "N"
5290      DISP "Insert next disc in current MSU   <CONTINUE> "
5300      PAUSE
```

```
5310    CASE ELSE
5320      BEEP
5330      GOTO 5170
5340    END SELECT
5350  END SELECT
5360  RETURN
5370  !=============================================================================
5380  END
```

```
10    !   NMO_CORR  --  Normal moveout corrections  ---   no plotting routines
20    !
30    !   A.T.Dippenaar   ---   U.P.  ---   April 1987
40    !
50    !   Last changes on 11 January 1988
60    !=============================================================================
70    !
80    OPTION BASE 1
90    !
100   COM /Seis_int/ Line$[2],INTEGER Trace(24,960),Opt,Amt,Gain(24)
110   COM /Seis_real/ REAL Shot(24),Geo(24),Shift_value(24),Offset
120   COM /Processed/ REAL Processed_trace(24,959),Pointy,V
130   COM /Polar/ INTEGER Pol_factor(24)
140   COM /Statics/ REAL Geophone_static(100),Source_static(100),Peg(100)
150   COM /Screens/ INTEGER Screen_1(7500),Screen_2(7500),REAL X,Y,Gain_amplitud
e,Gain_time
160   !
170   !=============================================================================
180   INTEGER Dummy(961),Mute(25),Store_trace(959),Cmp_store(959),Sample_shift(4
5)
190   DIM Twt(8),Input_vel(8),Stack_vel(959),Nmo_cor_trace(24,959),Cmp_stacked(9
59),Nmo(24,959)
200   DIM Marker$[6]
210   REAL Difference,Diff
220   Marker$="==>"&CHR$(8)&CHR$(8)&CHR$(8)
230   !
240   CONTROL 1,12;1
250   OUTPUT KBD;"K";
260   Sample_interval=192/959                ! Sample interval in msec
270   Si=Sample_interval/1000                ! Sample interval in sec
280   Sweep=192
290   !
300   Ques$="N"
310   DISP "Ensure disc for output files in left hand drive ( '':,700,0'' )   <C
ONTINUE> "
320   PAUSE
330   INPUT "Line identification (Maximum 2 alphanumericals)",Line$
340   !
350   Cdp_fold=6
360   INPUT "Fold of proposed CDP stack [ DEF = 6 ]",Cdp_fold
370   Zero_trace_ques$="N"
380   INPUT "Must the zero offset trace be included in the stack [ DEF = N ]",Ze
ro_trace_ques$
390   Amt=Cdp_fold
400   Shot_space=5
410   INPUT "Shot spacing  [ DEF = 5 ]",Shot_space
420   INPUT "First CDP midpoint (shotpoint location)",First_cdp
430   INPUT "Last CDP midpoint (shotpoint location)",Last_cdp
440   !
450   !
460   Stat_ques$="Y"
470   INPUT "Must static correction be applied [ DEF = Y ]",Stat_ques$
480   !
490   IF Stat_ques$="Y" THEN
500     !
510     INPUT "First peg position",First_peg
520     INPUT "Last peg position",Last_peg
530     Peg_space=5
540     INPUT "Peg spacing [ DEF = 5 ]",Peg_space
550     Peg_amt=(Last_peg-First_peg)/Peg_space+1
```

```
560     FOR P=1 TO Peg_amt
570       Peg(P)=First_peg+(P-1)*Peg_space
580     NEXT P
590     !
600     MASS STORAGE IS ":,700,1
610     !
620     Fil$=Line$&"_srce_st"
630     ASSIGN @P TO Fil$
640     ON END @P GOTO 660
650     ENTER @P;Source_static(*)
660     ASSIGN @P TO *
670     !
680     Fil$=Line$&"_geo_st"
690     ASSIGN @P TO Fil$
700     ON END @P GOTO 720
710     ENTER @P;Geophone_static(*)
720     ASSIGN @P TO *
730     !
740   END IF
750   !
760   FOR Tr=1 TO Amt
770     DISP "Enter mute (in msec) for trace";Tr;
780     INPUT Mute_time
790     Mute(Tr)=INT(Mute_time*959/192)
800   NEXT Tr
810   Mute(Amt+1)=959
820   !
830   !=============================================================================
840   PRINT TABXY(14,14);"  First TWT-velocity pair must be at time = 0 msec"
850   PRINT TABXY(14,16);"  Last TWT-velocity pair must be at time = 192 msec"
860   !
870   INPUT "How many TWT -  Velocity pairs are you entering",Nr_pairs
880   IF Nr_pairs>8 THEN
890     BEEP
900     DISP "Let's be reasonable   --  Try for less than 8 please"
910     BEEP
920     WAIT 1.5
930     GOTO 870
940   END IF
950   !
960   DISP "TWT,velocity pair # 1";
970   INPUT Twt(1),Input_vel(1)
980   IF Twt(1)<>0 THEN 930
990   !
1000  OUTPUT KBD;"K";
1010  PRINT TABXY(20,5);" TWT          !    Stacking velocity"
1020  PRINT TABXY(20,6);"-------------------------------------"
1030  PRINT TABXY(33,7);"!"
1040  PRINT TABXY(21,7);Twt(1)
1050  PRINT TABXY(40,7);Input_vel(1)
1060  !
1070  FOR I=2 TO Nr_pairs
1080    DISP "TWT,velocity pair # ";I;
1090    INPUT Twt(I),Input_vel(I)
1100    PRINT TABXY(33,6+I);"!"
1110    PRINT TABXY(21,6+I);Twt(I)
1120    PRINT TABXY(40,6+I);Input_vel(I)
1130  NEXT I
1140  !
1150  IF Twt(I-1)<>192 THEN
```

```
1160    BEEP
1170    DISP "Last TWT is  not equal to 192 msec  --  Redo TWT,VELOCITY input"
1180    BEEP
1190    WAIT 1.5
1200    OUTPUT KBD;"K";
1210    GOTO 870
1220  END IF
1230  OUTPUT KBD;"K";
1240  !=============================================================================
1250  !
1260  !   Calculating stacking velocity for every sample point.  Interpolation
1270  !   to fill velocity values between input velocities.
1280  !
1290  !-----------------------------------------------------------------------------
1300  PRINT TABXY(23,9);"Interpolating stacking velocities"
1310  FOR V=1 TO Nr_pairs-1
1320    Diff=Sample_interval*(Input_vel(V+1)-Input_vel(V))/(Twt(V+1)-Twt(V))
1330    Index1=Twt(V)/Sample_interval
1340    IF Index1=0 THEN Index1=1
1350    Index2=Twt(V+1)/Sample_interval
1360    Stack_vel(Index1)=Input_vel(V)                          !   in m/s
1370    FOR Vv=Index1+1 TO Index2
1380      Stack_vel(Vv)=Stack_vel(Vv-1)+Diff
1390    NEXT Vv
1400  NEXT V
1410  Stack_vel(959)=Input_vel(Nr_pairs)
1420  !=============================================================================
1430  !
1440  FOR Cdp=First_cdp TO Last_cdp STEP Shot_space
1450    OUTPUT KBD;"K";
1460    !
1470    Position=Cdp
1480    FOR Tr=1 TO Amt
1490      Shot(Tr)=Cdp-((Tr-1)*Shot_space)
1500      Geo(Tr)=Cdp+((Tr-1)*Shot_space)
1510    NEXT Tr
1520    !
1530    PRINT TABXY(28,9);"Gathering CMP traces"
1540    Ms$=":,700,0,2"
1550    MASS STORAGE IS Ms$
1560    FOR Tr=1 TO Amt
1570      ON ERROR GOSUB To_err_is_human
1580      GOSUB Read_trace
1590      OFF ERROR
1600    NEXT Tr
1610    Max_offset=ABS(Shot(Amt)-Geo(Amt))
1620  !
1630  !=============================================================================
1640    IF Stat_ques$="Y" THEN
1650      FOR Tr=1 TO Amt
1660        PRINT TABXY(22,9);"Applying static corrections  --  trace ";Tr
1670        Total_static(Tr)=0
1680        FOR P=1 TO Peg_amt
1690          IF Shot(Tr)=Peg(P) THEN Total_static(Tr)=Total_static(Tr)+Source_s
tatic(P)
1700          IF Geo(Tr)=Peg(P) THEN Total_static(Tr)=Total_static(Tr)+Geophone_
static(P)
1710        NEXT P
1720        Sample_shift(Tr)=INT(Total_static(Tr)/Si)
1730      !
```

```
1740      !            Applying static corrections
1750      !
1760        FOR I=1 TO 959-Sample_shift(Tr)
1770          Trace(Tr,I)=Trace(Tr,I+Sample_shift(Tr))
1780        NEXT I
1790      !
1800        FOR I=959-Sample_shift(Tr) TO 959
1810          Trace(Tr,I)=0
1820        NEXT I
1830      !
1840      NEXT Tr
1850    END IF
1860  !
1870  !=============================================================================
1880    FOR Tr=1 TO Amt
1890      PRINT TABXY(20,9);"Generating NMO values :-- offset ";Tr;"
    "
1900    !
1910      Seperation=ABS(Geo(Tr)-Shot(Tr))
1920      FOR I=1 TO 959
1930        T0=(I*Sample_interval)^2
1940        V=Stack_vel(I)/1000                          !   m/msec
1950        Nmo(Tr,I)=SQR(T0+Seperation^2/V^2)
1960      NEXT I
1970    !
1980    NEXT Tr
1990  !
2000  !=============================================================================
2010  !   Shifting values to NMO corrected sample positions.
2020  !=============================================================================
2030    OUTPUT KBD;"K";
2040    FOR Tr=1 TO Amt
2050      PRINT TABXY(26,9);"Doing NMO correction ";Tr
2060      FOR I=1 TO 959
2070        Remainder=FRACT(Nmo(Tr,I)/Sample_interval)*Sample_interval
2080        T1=Nmo(Tr,I)-Remainder
2090        T2=T1+Sample_interval
2100        Index1=T1/Sample_interval
2110        Index2=T2/Sample_interval
2120      !-------------------------------------------------------------------
2130      !     T1,T2 are TWT's (multiples of sample interval) which
2140      !     straddles calculated NMO correction. Index1 and Index2
2150      !     are corresponding sample numbers
2160      !
2170      !     'Difference' is interpolated value between T1 and T2
2180      !-------------------------------------------------------------------
2190        IF Index1>958 THEN 2350
2200        ON ERROR GOTO 2230
2210        Diff=Trace(Tr,Index2)-Trace(Tr,Index1)
2220        GOTO 2330
2230        IF ERRN=20 THEN
2240          OFF ERROR
2250          Diff=0
2260          GOTO 2320
2270        ELSE
2280          BEEP
2290          DISP "ooops ----  ERRN =";ERRN
2300          PAUSE
2310        END IF
2320        OFF ERROR
```

```
2330        Difference=Diff*(Remainder/Sample_interval)
2340        Nmo_cor_trace(Tr,I)=Trace(Tr,Index1)+Difference
2350      NEXT I
2360      PRINT TABXY(15,9);"
2370    !
2380    NEXT Tr
2390  !=============================================================================
2400 Mute_and_stack: !
2410  !-----------------------------------------------------------------------------
2420  !  Muting routine
2430  !-----------------------------------------------------------------------------
2440    OUTPUT KBD;"K";
2450    FOR Tr=1 TO Amt
2460      PRINT TABXY(32,9);"Muting trace ";Tr
2470      FOR M=1 TO Mute(Tr)
2480        Nmo_cor_trace(Tr,M)=0
2490      NEXT M
2500    NEXT Tr
2510  !-----------------------------------------------------------------------------
2520  !  Stacking routine
2530  !-----------------------------------------------------------------------------
2540    PRINT TABXY(31,9);"Stacking trace
2550    Begin=2
2560    IF Zero_trace_ques$="Y" THEN Begin=1
2570    FOR Tr=Begin TO Amt
2580      FOR S=1 TO 959
2590        Cmp_stacked(S)=Cmp_stacked(S)+Nmo_cor_trace(Tr,S)
2600      NEXT S
2610    NEXT Tr
2620  !-----------------------------------------------------------------------------
2630  !   Fold compensation  and write to INTEGER format
2640  !-----------------------------------------------------------------------------
2650    Tr=1
2660    !
2670    FOR I=1 TO 959
2680      Cmp_temp=INT(Cmp_stacked(I)/SQR(Tr))
2690      IF I>Mute(Tr) THEN Tr=Tr+1
2700      IF Cmp_temp>32767 THEN Cmp_temp=32767
2710      IF Cmp_temp<-32766 THEN Cmp_temp=-32766
2720      Cmp_store(I)=Cmp_temp
2730    NEXT I
2740    !
2750    FOR I=1 TO 959
2760      Cmp_stacked(I)=0
2770    NEXT I
2780  !-----------------------------------------------------------------------------
2790  !   Storing CMP stacked trace on disc
2800  !-----------------------------------------------------------------------------
2810    MASS STORAGE IS ":,700,0"
2820    Fil$=Line$&"_"&VAL$(Position)
2830    DISP Fil$
2840    CREATE BDAT Fil$,1,1950
2850    ASSIGN @To_disc TO Fil$
2860    ON END @To_disc GOTO 2880
2870    OUTPUT @To_disc;Cmp_store(*)
2880    ASSIGN @To_disc TO *
2890  !
2900  NEXT Cdp
2910  MASS STORAGE IS ":,700,0  "
2920  LOAD "AUTOST"
```

```
2930  STOP
2940  !-------------------------------------------------------------------------
2950 Read_trace:!
2960  !
2970  MASS STORAGE IS ":,700,1"
2980  DISP TAB(46);"Reading trace ";Tr;" of ";Amt
2990  IF Shot(Tr)<0 THEN
3000    Fil$=Line$&"n"&VAL$(ABS(Shot(Tr)))&"_"&VAL$(Geo(Tr))
3010  ELSE
3020    Fil$=Line$&"_"&VAL$(Shot(Tr))&"_"&VAL$(Geo(Tr))
3030  END IF
3040  !
3050  ON ERROR GOSUB To_err_is_human
3060  ASSIGN @To_disc TO Fil$
3070  ON END @To_disc GOTO 3090
3080  ENTER @To_disc;Dummy(*)
3090  FOR I=1 TO 959
3100    Trace(Tr,I)=Dummy(I)
3110  NEXT I
3120  Gain(Tr)=Dummy(960)
3130  ASSIGN @To_disc TO *
3140  OFF ERROR
3150  !
3160  DISP "
        "
3170  RETURN
3180  !-------------------------------------------------------------------------
3190 To_err_is_human: !
3200  SELECT ERRN
3210  CASE 56
3220    Ms_counter=Ms_counter+1
3230    !
3240    SELECT Ms$
3250    CASE ":,700,0,2"
3260      Ms$=":,700,0,3"
3270    CASE ":,700,0,3"
3280      Ms$=":,700,0,2"
3290    END SELECT
3300    MASS STORAGE IS Ms$
3310    !
3320    IF Ms_counter>3 THEN
3330      BEEP
3340      Ms_counter=0
3350      PRINT TABXY(20,14);" Couldn't find ";Fil$;" on current MSU"
3360      INPUT "Do you want to change the MSU",Msu_ques$
3370      SELECT Msu_ques$
3380      CASE "Y"
3390        INPUT "Enter new MSU  - e.g. '':,700,0,3'' [ NO DEF ]",Ms$
3400        ON ERROR GOTO 3440
3410        MASS STORAGE IS Ms$
3420        OFF ERROR
3430        GOTO 3470
3440        BEEP
3450        DISP "TRY AGAIN Bimbo ----";
3460        GOTO 3390
3470      CASE "N"
3480        DISP "Insert next disc in current MSU   <CONTINUE> "
3490        PAUSE
3500      CASE ELSE
3510        BEEP
```

```
3520         GOTO 3360
3530       END SELECT
3540     END IF
3550   END SELECT
3560   RETURN
3570   !=============================================================================
3580   END
```

```
10    !   FILTR_BUTR  --  Butterworth filter program
20    !   Adapted from BUTFIL FORTRAN program by J.Fatti -- PhD thesis CSM
30    !   A.T.Dippenaar   ---   U.P  ---   April 1987
40    !
50    !   Last changes on 15 January 1988
60    !=============================================================================
70    !
80    OPTION BASE 1
90    !
100   COM /Seis_int/ Line$[2],INTEGER Trace(24,960),Opt,Amt,Gain(24)
110   COM /Seis_real/ REAL Shot(24),Geo(24),Shift_value(24),Offset
120   COM /Processed/ REAL Processed_trace(24,959),Pointy,V
130   COM /Polar/ INTEGER Pol_factor(24)
140   COM /Statics/ REAL Geophone_static(100),Source_static(100),Peg(100)
150   COM /Screens/ INTEGER Screen_1(7500),Screen_2(7500),REAL X,Y,Gain_amplitud
e,Gain_time
160   !
170   !=============================================================================
180   !
190   DIM Coeff(5,7),Fins(1163),Outs(1163),C(5),Fin(1163),Out(1163)
200   INTEGER Int_trace(959)
210   OUTPUT KBD;"K";
220   !
230   INPUT "Do you wish to store the filtered traces (Y/N)",Store_ques$
240   INPUT "Enter filter passband (low cut,high cut)",F1,F2
250   INPUT " Order of butterworth filter (must be even) ",Order
260   IF INT(Order/2)*2<>Order THEN
270     BEEP
280     GOTO 250
290   END IF
300   N=Order
310   !
320   Sample_interval=192/959                     ! in miliseconds
330   Delt=Sample_interval/1000                   ! in seconds
340   !-----------------------------------------------------------------------
350   !   Calculate Butterworth coefficients
360   !-----------------------------------------------------------------------
370   FOR M=1 TO Order/2
380     GOSUB Butter_coeff
390     Coeff(M,1)=1/C(1)
400     Coeff(M,2)=-2/C(1)
410     Coeff(M,3)=1/C(1)
420     Coeff(M,4)=C(2)/C(1)
430     Coeff(M,5)=C(3)/C(1)
440     Coeff(M,6)=C(4)/C(1)
450     Coeff(M,7)=C(5)/C(1)
460   NEXT M
470   !***************************************************************************
480   FOR Tr=1 TO Amt
490     DISP "Filtering trace ";Tr;" of ";Amt;"    ---  input order"
500     !
510     FOR J=1 TO 4
520       Fin(J)=0
530       Out(J)=0
540     NEXT J
550     !
560     FOR J=5 TO 959+4
570       Fin(J)=Trace(Tr,J-4)
580     NEXT J
590     !
```

```
600     FOR J=959+4 TO 959+200
610       Fin(J)=0
620     NEXT J
630     !
640     FOR M=1 TO Order/2
650       IF M>1 THEN                ! Set input = previous output
660         FOR K=5 TO 959+200
670           Fin(K)=Out(K)
680         NEXT K
690       END IF
700       !-------------------------------------------------------------------
710       !   Choose appropriate coefficients
720       !-------------------------------------------------------------------
730       Fn1=Coeff(M,1)
740       Fn2=Coeff(M,2)
750       Fn3=Coeff(M,3)
760       D1=Coeff(M,4)
770       D2=Coeff(M,5)
780       D3=Coeff(M,6)
790       D4=Coeff(M,7)
800       !-------------------------------------------------------------------
810       !   Filter array FIN  recursively with one polynomial fraction to
820       !   get array OUT
830       !-------------------------------------------------------------------
840       FOR I=5 TO 959+200
850         Fnumin=Fn1*Fin(I)+Fn2*Fin(I-2)+Fn3*Fin(I-4)
860         Denout=D1*Out(I-1)+D2*Out(I-2)+D3*Out(I-3)+D4*Out(I-4)
870         Out(I)=Fnumin-Denout   .
880       NEXT I
890     NEXT M
900     !-------------------------------------------------------------------
910     !  Set the first 4 input and output samples = 0
920     !-------------------------------------------------------------------
930     FOR J=1 TO 4
940       Fins(J)=0
950       Outs(J)=0
960     NEXT J
970     !
980     DISP "Filtering trace ";Tr;" of ";Amt;"   ---  reverse order"
990     !
1000    FOR M=1 TO Order/2
1010      IF M=1 THEN                ! Reverse previous output
1020        FOR I=1 TO 959+196
1030          Fins(I)=Out(959+201-I)
1040        NEXT I
1050      ELSE                       ! Set input = previous output
1060        FOR K=5 TO 959+200
1070          Fins(K)=Outs(K)
1080        NEXT K
1090      END IF
1100      !-------------------------------------------------------------------
1110      !    Choose appropriate coefficients
1120      !-------------------------------------------------------------------
1130      FOR L=1 TO 3
1140        Fn(L)=Coeff(M,L)
1150      NEXT L
1160      FOR L=1 TO 4
1170        D(L)=Coeff(M,L+3)
1180      NEXT L
1190      !-------------------------------------------------------------------
```

```
1200      !  Filter array FINS ( output of previous filter ) recursively
1210      !  with one polynomial fraction to get array OUTS
1220      !-------------------------------------------------------------------
1230      FOR I=5 TO 959+200
1240        Fnumin=Fn(1)*Fins(I)+Fn(2)*Fins(I-2)+Fn(3)*Fins(I-4)
1250        Denout=0
1260        FOR J=1 TO 4
1270          Aa=D(J)*Outs(I-J)
1280          Denout=Denout+Aa
1290        NEXT J
1300        Outs(I)=Fnumin-Denout
1310      NEXT I
1320    NEXT M
1330    !-------------------------------------------------------------------
1340    !  Reverse array OUTS and put into array FIN. The filtered time series
1350    !  is now the right way round
1360    !-------------------------------------------------------------------
1370    FOR I=1 TO 959
1380      Fin(I)=Outs(959+201-I)
1390    NEXT I
1400    !-------------------------------------------------------------------
1410    !   Amplitude normalization
1420    !-------------------------------------------------------------------
1430    DISP "Normalizing trace ";Tr;" of ";Amt
1440    !
1450    First=120               !  Calculating trace mean from sample 120
1460    Last=959                !  Ending at 959
1470    Sum=0
1480    FOR I=First TO Last
1490      Sum=ABS(Fin(I))+Sum
1500    NEXT I
1510    Trace_mean=Sum/(Last-First)
1520    !
1530    Ref_amplitude=12000
1540    !
1550    FOR I=1 TO 959
1560      Processed_trace(Tr,I)=Fin(I)*Ref_amplitude/Trace_mean
1570      IF ABS(Processed_trace(Tr,I))>32766 THEN
1580        IF Processed_trace(Tr,I)>32766 THEN Int_trace(I)=32766
1590        IF Processed_trace(Tr,I)<-32766 THEN Int_trace(I)=-32766
1600      ELSE
1610        Int_trace(I)=INT(Processed_trace(Tr,I))
1620      END IF
1630    NEXT I
1640    !-------------------------------------------------------------------
1650    !   Writing processed trace to disc
1660    !-------------------------------------------------------------------
1670    !
1680    IF Store_ques$="Y" THEN
1690      MASS STORAGE IS ":,700,1"
1700      Fil$=Line$&VAL$(Shot(Tr))&"f"&VAL$(Geo(Tr))
1710      CREATE BDAT Fil$,1,1920
1720      ASSIGN @Out TO Fil$
1730      OUTPUT @Out;Int_trace(*)
1740      ASSIGN @Out TO *
1750    END IF
1760    !
1770  NEXT Tr
1780  !
1790  LOAD "AUTOST:,700"
```

```
1800  STOP
1810  !=============================================================================
1820 Butter_coeff: !                          JORGE PARRA - CSM - APRIL 1972
1830               !
1840  F2a=SIN(PI*F2*Delt)/COS(PI*F2*Delt)
1850  F2a=F2a/(PI*Delt)
1860  F1a=SIN(PI*F1*Delt)/COS(PI*F1*Delt)
1870  F1a=F1a/(PI*Delt)
1880  W0=2*PI*SQR(F1a*F2a)
1890  Wc=2*PI*(F2a-F1a)
1900  Q=W0/Wc
1910  Del=Wc*Delt
1920  U=PI/(2*N)
1930  K=N+2*M-1
1940  B21=-2*COS(U*K)
1950  X=2/Del
1960  Y=X*X
1970  Z=X*B21
1980  U=2*Q*Q+1
1990  V=B21*Q*Q/X
2000  W1=Q^4
2010  W=W1/Y
2020  C(5)=Y-Z+U-V+W
2030  C(4)=(-4*Y)+(2*Z)-(2*V)+(4*W)
2040  C(3)=(6*Y)-(2*U)+(6*W)
2050  C(2)=(-4*Y)-(2*Z)+(2*V)+(4*W)
2060  C(1)=Y+Z+U+V+W
2070  RETURN
2080  !
2090  END
```

```
10     !  RAY_TRACE   ---   Ray tracing routine
20     !
30     !  A.T.Dippenaar   ---   U.P   ---   March 1986
40     !
50     !  Last changes on 6 APRIL 1986
60     !*****************************************************************************
70     !
80     OPTION BASE 0
90     DIM Geo(25),Total_time(25,10)
100    INTEGER Screen(7500),Model(7500)
110    PRINT CHR$(12)
120    DEG
130    PLOTTER IS 3,"INTERNAL"
140    GINIT
150    GCLEAR
160    GRAPHICS OFF
170    !
180    INPUT "Nr of geophones",Amt
190    INPUT "Geophone 1 -- offset",Gp
200    IF Amt<>1 THEN INPUT "Geophone spacing",Space
210    FOR I=1 TO Amt
220      Geo(I)=Gp+(I-1)*Space
230    NEXT I
240    IF Space=0 THEN Space=100
250    Max_offset=Geo(Amt)+Space
260    INPUT "What accuracy (in metres) do you require at geophone positions",Acc
uracy
270    !
280    INPUT "Depth to last interface",Max_depth
290    Xx=133
300    Yy=100
310    VIEWPORT .1*Xx,Xx,.1*Yy,.85*Yy
320  ! MOVE 0,.85*Yy
330  ! DRAW 133,.85*Yy
340    WINDOW 0,Max_offset,-Max_depth,0
350    MOVE 0,0
360    DRAW Max_offset,0
370    CLIP OFF
380    LORG 4
390    FOR I=1 TO Amt
400      MOVE Geo(I),0
410      LABEL "V"
420    NEXT I
430    LORG 8
440    CSIZE 2.8
450    FOR I=-Max_depth TO 0 STEP 1000
460      MOVE 0,I
470      LABEL -I
480    NEXT I
490    CLIP ON
500    !
510    Nr_interface=0
520    DISP "Interface 1"
530    GOTO Next_interface
540    !
550    ON KEY 5 LABEL "Next interface" GOTO Next_interface
560    ON KEY 6 LABEL "                " GOTO Spinner
570    ON KEY 7 LABEL "CONTINUE" GOTO Continue
580    ON KEY 8 LABEL "                " GOTO Spinner
590    ON KEY 9 LABEL "Undo last" GOTO Undo
```

```
600   FOR K=0 TO 4
610     ON KEY K LABEL "    " GOTO Spinner
620   NEXT K
630 Spinner:GOTO Spinner
640   !
650 Next_interface:    !
660   GRAPHICS OFF
670   Nr_interface=Nr_interface+1
680   DISP "Interface ";Nr_interface
690   ON KEY 5 LABEL "Horizontal " GOTO Horizontal
700   ON KEY 6 LABEL "Up dip   " GOTO Up_dip
710   ON KEY 7 LABEL "Down dip   " GOTO Down_dip
720   ON KEY 8 LABEL "               " GOTO Spin
730   ON KEY 9 LABEL "ESCAPE" GOTO 740
740 Spin:GOTO Spin
750   !
760 Undo:  !
770   IF Undo_flag$<>"Set" THEN
780     GLOAD Screen(*)
790     Nr_interface=Nr_interface-1
800   END IF
810   Undo_flag$="Set"
820   GOTO 550
830   !
840 Horizontal:   !
850   Undo_flag$="clear"
860   GSTORE Screen(*)
870   DISP "Depth to interface ";Nr_interface;
880   INPUT Depth_interface(Nr_interface)
890   Depth_interface(Nr_interface)=Depth_interface(Nr_interface)*(-1)
900   Dip(Nr_interface)=0
910   GOTO 1130
920   !
930 Up_dip:  !
940   Undo_flag$="clear"
950   GSTORE Screen(*)
960   DISP "Depth to left side of interface ";Nr_interface;
970   INPUT Depth_interface(Nr_interface)
980   DISP "Dip of interface ";Nr_interface;
990   INPUT Dip(Nr_interface)
1000  Depth_interface(Nr_interface)=Depth_interface(Nr_interface)*(-1)
1010  GOTO 1130
1020  !
1030 Down_dip: !
1040  Undo_flag$="clear"
1050  GSTORE Screen(*)
1060  DISP "Depth to left side of interface ";Nr_interface;
1070  INPUT Depth_interface(Nr_interface)
1080  DISP "Dip of interface ";Nr_interface;
1090  INPUT Dip(Nr_interface)
1100  Dip(Nr_interface)=Dip(Nr_interface)*(-1)
1110  Depth_interface(Nr_interface)=Depth_interface(Nr_interface)*(-1)
1120  !
1130  GRAPHICS ON
1140  MOVE 0,Depth_interface(Nr_interface)
1150  DRAW Max_offset,Depth_interface(Nr_interface)+(Max_offset*TAN(Dip(Nr_inter
face)))
1160  !
1170  DISP " Velocity of layer ";Nr_interface;
1180  INPUT V(Nr_interface)
```

```
1190  CSIZE 3
1200  LORG 8
1210  IF Nr_interface=1 THEN
1220    MOVE Max_offset,Depth_interface(1)/2
1230  ELSE
1240    MOVE Max_offset,(Depth_interface(Nr_interface)-Depth_interface(Nr_interf
ace-1))/2+Depth_interface(Nr_interface-1)
1250  END IF
1260  LABEL V(Nr_interface)
1270  !
1280  GOTO 550
1290  !
1300 Continue: !
1310  GSTORE Screen(*)
1320  GSTORE Model(*)
1330! GOSUB Print_head
1340  DEG
1350  L=0
1360  Depth_interface(0)=0
1370  LOOP
1380    L=L+1
1390    Transmitted_d(0)=5
1400    Increment=.3
1410    G=1
1420  !
1430    Dip(0)=0
1440    LOOP
1450    !
1460      FOR K=0 TO 9
1470        ON KEY K LABEL "               " GOTO 1590
1480      NEXT K
1490    !
1500      ON KEY 5 LABEL "CNGE INC,REDO" GOTO Redo
1510      GOTO 1590
1520 Redo:   !
1530      GLOAD Model(*)
1540      GSTORE Screen(*)
1550      Transmitted_d(0)=5
1560      Increment=Increment-.05
1570      G=1
1580    !
1590      Transmitted_d(0)=Transmitted_d(0)+Increment
1600  !^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
1610  !   Downwards travelling wavefront   ---  Computation
1620  !^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
1630      IF L>Nr_interface THEN 2500
1640      FOR I=1 TO L
1650        Incident_d(I)=Transmitted_d(I-1)+Dip(I-1)-Dip(I)
1660    !
1670        IF I=L OR V(I+1)*SIN(Incident_d(I))/V(I)>=1 THEN
1680          Reflected(I)=Incident_d(I)
1690          Last=I
1700          GOTO 1780
1710        END IF
1720    !
1730        Transmitted_d(I)=ASN(V(I+1)*SIN(Incident_d(I))/V(I))
1740      NEXT I
1750  !^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
1760  !    Reflected wavefront    ----    Computation
1770  !^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

```
1780      Transmitted_u(I)=Reflected(I)
1790      FOR J=Last TO 1 STEP -1
1800        Incident_u(J-1)=Transmitted_u(J)-Dip(J)+Dip(J-1)
1810        IF J=1 THEN 1870
1820        Transmitted_u(J-1)=ASN(V(J-1)*SIN(Incident_u(J-1))/V(J))
1830      NEXT J
1840  !^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
1850  !     Draw downgoing wavefront
1860  !^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
1870      MOVE 0,0
1880      Shft_x=0
1890      Shft_y=0
1900      FOR I=1 TO L
1910        Angle_1=90-Incident_d(I)
1920        Angle_2=90-Dip(I)
1930        Angle_3=Transmitted_d(I-1)+Dip(I-1)
1940        Side_1=-Depth_interface(I)-Shft_y-(Shft_x*TAN(Dip(I)))
1950        Side_2=SIN(Angle_2)*(Side_1/SIN(Angle_1))
1960        Side_3=SIN(Angle_3)*(Side_1/SIN(Angle_1))
1970        Shft_x_1=Side_3*COS(Dip(I))
1980        Shft_x=Shft_x_1+Shft_x
1990        Shft_y_1=(-1)*(Side_3*SIN(Dip(I)))
2000        Shft_y=Side_1+Shft_y_1+Shft_y
2010  !
2020        Path_length_d(I)=Side_2
2030        DRAW Shft_x,-Shft_y
2040  !
2050      NEXT I
2060  !^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
2070  !  Draw reflected wavefront and "freezes" wavefront at geophone positions
2080  !^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
2090      FOR I=L TO 1 STEP -1
2100        Angle_1=90-Incident_u(I-1)
2110        Angle_2=90+Dip(I-1)
2120        Angle_3=Transmitted_u(I)-Dip(I)
2130        Side_1=Shft_y+Depth_interface(I-1)+(Shft_x*TAN(Dip(I-1)))
2140        Side_2=SIN(Angle_2)*(Side_1/SIN(Angle_1))
2150        Side_3=SIN(Angle_3)*(Side_1/SIN(Angle_1))
2160        Shft_x_1=Side_3*COS(Dip(I-1))
2170        Shft_x=Shft_x_1+Shft_x
2180        Shft_y_1=Side_3*SIN(Dip(I-1))
2190        Shft_y=Shft_y-Shft_y_1-Side_1
2200        !
2210        Path_length_u(I)=Side_2
2220        DRAW Shft_x,-Shft_y
2230        !
2240        IF I=1 THEN
2250          IF Shft_x>Max_offset THEN GOTO Redo    !  Missed last geophone
2260          FOR K=G TO Amt
2270            IF Shft_x>Geo(K)-Accuracy AND Shft_x<Geo(K)+Accuracy THEN
2280              GOSUB Calculate
2290              IF G=Amt THEN 2410
2300              G=G+1
2310              GSTORE Screen(*)
2320              GOTO 2390
2330            END IF
2340          NEXT K
2350          GOTO 2370
2360        END IF
2370      NEXT I
```

```
2380
2390      GLOAD Screen(*)
2400    END LOOP
2410    GSTORE Screen(*)
2420    GSTORE Model(*)
2430    IF L=Nr_interface THEN 2450
2440  END LOOP
2450! GOSUB Print_times
2460  !
2470  STOP
2480  !=============================================================================
2490 Print_head: !
2500  IMAGE 19X,DDDDD,AA,10X,MDD,10X,DDDD,AAAA
2510! OUTPUT 701;CHR$(27)&"(s1B"&"SEISMIC MODELING :  "&CHR$(10)
2520! OUTPUT 701;CHR$(27)&"&dD"&"Model :"&CHR$(27)&"&d@"&"                Depth
        Dip          Velocity"&CHR$(27)&"(s0B"
2530  FOR P=1 TO Nr_interface
2540    OUTPUT 701 USING 2500;-Depth_interface(P)," m",Dip(P),V(P)," m/s"
2550  NEXT P
2560  OUTPUT 701;CHR$(10),CHR$(10)
2570  RETURN
2580  !=============================================================================
2590 Print_times: !
2600  IMAGE X,DDDD,3X,A,2X,#
2610  IMAGE D.DDDD,4X,#
2620  OUTPUT 701;CHR$(27)&"(s1B"&"REFLECTION TIMES :  "&CHR$(10)
2630  OUTPUT 701;CHR$(27)&"(s1B"&"Offset  !  ";
2640  FOR I=1 TO Nr_interface
2650    OUTPUT 701;"Layer";I;"    ";
2660  NEXT I
2670  OUTPUT 701;CHR$(10);CHR$(27)&"(s0B"
2680  FOR Ge=1 TO Amt
2690    OUTPUT 701;CHR$(27)&"(s1B";
2700    OUTPUT 701 USING 2600;Geo(Ge);"!  "
2710    OUTPUT 701;CHR$(27)&"(s0B";
2720    FOR La=1 TO Nr_interface
2730      OUTPUT 701 USING 2610;Total_time(Ge,La)
2740    NEXT La
2750    OUTPUT 701;CHR$(10)
2760  NEXT Ge
2770  OUTPUT 701;CHR$(12)
2780  RETURN
2790  !=============================================================================
2800 Calculate:  !
2810  Total_time_u=0
2820  Total_time_d=0
2830  FOR T=1 TO L
2840    Time_d(T)=Path_length_d(T)/V(T)
2850    Total_time_d=Total_time_d+Time_d(T)
2860  NEXT T
2870  !
2880  FOR T=L TO 1 STEP -1
2890    Time_u(T)=Path_length_u(T)/V(T)
2900    Total_time_u=Total_time_u+Time_u(T)
2910  NEXT T
2920  !
2930  Total_time(G,L)=Total_time_u+Total_time_d
2940  !
2950  RETURN
2960  END
```

```
10    !   SYNTH_CMP   --  Synthetic CMP generating program
20    !
30    !   A.T.Dippenaar   ---   U.P.  ---   January 1988
40    !
50    !   Last changes on 10 January 1988
60    !=============================================================================
70    !
80    OPTION BASE 1
90    !
100   INTEGER Trace(24,959),Amt,Dum(959)
110   DIM Geo(24),T0(15),V(15),Event(24,15),Sample(24,15),Signal(80),Shot(24)
120   !
130   !=============================================================================
140   !
150   OUTPUT KBD;"K";
160   !
170   INPUT "Enter random seed (any number 10 to 5000)",Rs
180   !
190   FOR R=1 TO Rs           ! Just to ensure random noise is really random
200     Rnr=RND               ! for different CMP gathers
210   NEXT R
220   !
230   INPUT "Two letter identifier ",F$
240   Sweep=192
250   INPUT "Sweep length of synthetic record (msec)",Sweep
260   Sample_interval=Sweep/959               ! Sample interval in msec
270   Sample_inverse=959/Sweep
280   Si=Sample_interval/1000                 ! Sample interval in sec
290   !
300   !=============================================================
310   !
320   !   Setting up synthetic wavelet
330   !-----------------------------+--------------+-----------------------
340   !
350   Wavelet_length=INT(12*Sample_inverse)+1
360   FOR W=1 TO Wavelet_length
370     Time=(W-1)/Sample_inverse-2
380     Signal(W)=2*SIN(PI*Time/2)/(PI*Time)
390   NEXT W
400   !-------------------------------------------------------------------------
410   Lx=959                    !  convolution parameters  .... CLAERBOUT
420   Lb=Wavelet_length
430   Ly=Lx+Lb-1
440   !
450   ALLOCATE X(Lx),B(Lb),Y(Ly)
460   !
470   !=============================================================
480   INPUT "Fold of CMP     ",Amt
490   INPUT "CMP",Cmp
500   Sta_space=5
510   INPUT "Station spacing [ DEF = 5 ]",Sta_space
520   FOR Tr=1 TO Amt
530     Geo(Tr)=Cmp+(Tr-1)*Sta_space
540     Shot(Tr)=Cmp-(Tr-1)*Sta_space
550     PRINT Geo(Tr)-Shot(Tr)
560   NEXT Tr
570   !
580   INPUT "How many events on record",Event_amt
590   Noise$="Y"
600   INPUT "Add random noise (Y/N)",Noise$
```

```
610   Cnvlve$="N"
620   INPUT "Convolve spiked record with wavelet (Y/N)",Cnvlve$
630   !
640   FOR E=1 TO Event_amt
650     DISP "T0 for event ";E;" (msec) ";
660     INPUT T0(E)
670     T0(E)=T0(E)/1000                              ! Event in seconds
680     DISP "Velocity for event ";E;" (m/s)";
690     INPUT V(E)
700   NEXT E
710   !
720   FOR E=1 TO Event_amt
730     FOR Tr=1 TO Amt
740       Event(Tr,E)=SQR(T0(E)^2+(Geo(Tr)-Shot(Tr))^2/V(E)^2)
750       Sample(Tr,E)=INT(Event(Tr,E)/Si)
760       Trace(Tr,Sample(Tr,E))=12000
770     NEXT Tr
780   NEXT E
790   !
800   FOR Tr=1 TO Amt
810     !-------------------------------------------------------------------
820     IF Noise$="Y" THEN
830       Nr=INT(120*RND)
840       FOR R=1 TO Nr
850         Rndm_sample=INT(959*RND)
860         Trace(Tr,Rndm_sample)=INT(14000*RND)-7000
870       NEXT R
880     END IF
890     !-------------------------------------------------------------------
900     IF Cnvlve$="Y" THEN
910       DISP "Convolving trace ";Tr;" --- this may take some time"
920       FOR C=1 TO Ly
930         Y(C)=0
940       NEXT C
950       FOR C=1 TO Lx
960         FOR Cc=1 TO Lb
970           Y(C+Cc-1)=Y(C+Cc-1)+Trace(Tr,C)*Signal(Cc)
980         NEXT Cc
990       NEXT C
1000        !
1010      FOR I=1 TO 959
1020        Trace(Tr,I)=INT(Y(I))
1030      NEXT I
1040        !
1050    END IF
1060    !-------------------------------------------------------------------
1070  NEXT Tr
1080    !
1090  FOR Tr=1 TO Amt
1100    FOR I=1 TO 959
1110      Dum(I)=Trace(Tr,I)
1120    NEXT I
1130    Filename$=F$&"_"&VAL$(Shot(Tr))&"_"&VAL$(Geo(Tr))
1140    DISP "Storing ";Filename$;" on disc"
1150    MASS STORAGE IS ":,700,1"
1160    CREATE BDAT Filename$,1,1950
1170    ASSIGN @P TO Filename$
1180    OUTPUT @P;Dum(*)
1190    ASSIGN @P TO *
1200  NEXT Tr
```

```
1210  MASS STORAGE IS ":,700"
1220  OUTPUT KBD;"K";
1230  LOAD "AUTOST"
1240  !
1250  !=============================================================================
1260  END
```

```
10    !   AMP_SPECTR  --   Calculating amplitude spectrum
20    !
30    !   Original filter program : E.H.Stettler
40    !
50    !   A.T.Dippenaar   ---   U.P   ---   January 1988
60    !
70    !   Last changes on  17 January 1988
80    !=================================================================
90    !
100   OPTION BASE 1
110   !
120   COM /Seis_int/ Line$[2],INTEGER Trace(24,960),Opt,Amt,Gain(24)
130   COM /Seis_real/ REAL Shot(24),Geo(24),Shift_value(24),Offset
140   COM /Processed/ REAL Processed_trace(24,959),Pointy,V
150   COM /Polar/ INTEGER Pol_factor(24)
160   COM /Statics/ REAL Geophone_static(100),Source_static(100),Peg(100)
170   COM /Screens/ INTEGER Screen_1(7500),Screen_2(7500),REAL X,Y,Gain_amplitud
e,Gain_time
180   !=================================================================
190   INTEGER I,N,Filtered_trace(959)
200   DIM N1(2048),P1(2048),Frek(2048),Nn1(2048)
210   DIM Filres(2048)
220   !
230   OUTPUT KBD;"K";
240   CONTROL 1,12;1
250   PLOTTER IS 3,"INTERNAL"
260   X=100*MAX(1,RATIO)
270   Y=100*MAX(1,1/RATIO)
280   GCLEAR
290   GINIT
300   GRAPHICS ON
310   N=1024
320   Sweep=1024*.192/959
330   !
340   PRINT TABXY(13,18);"There are ";Amt;" traces in common memory at this mome
nt"
350   INPUT "          What is the trace number of the data you wish to transfor
m",Tr
360   OUTPUT KBD;"K";
370   Format$="I"
380   INPUT "Format of input data  REAL (R) / INTEGER (I)   [ DEF = I ]",Format$
390   IF Format$<>"R" AND Format$<>"I" THEN 380
400   !
410   IF Format$="R" THEN
420     FOR I=1 TO 959
430       N1(2*I-1)=Processed_trace(Tr,I)
440       N1(2*I)=0
450     NEXT I
460   ELSE
470     FOR I=1 TO 959
480       N1(2*I-1)=Trace(Tr,I)*1.
490       N1(2*I)=0
500     NEXT I
510   END IF
520     !
530   CS=0.
540   S=-1
550   GOSUB Trace_plot
560   GOSUB Fft
570   OFF ERROR
```

```
580   GOSUB Amplitude
590   GOSUB Spectrum_plot
600   Dump_ques$="Y"
610   INPUT "Graphics dump of screen (Y/N) [DEF = Y]",Dump_ques$
620   IF Dump_ques$="Y" THEN
630     DUMP DEVICE IS 701
640     DUMP GRAPHICS
650   END IF
660   INPUT "Another amplitude spectrum calculation (Y/N)",Amp_ques$
670   IF Amp_ques$="Y" THEN 230
680   LOAD "AUTOST:,700"
690   STOP
700   !-------------------------------------------------------------------
710   !  FFT ROUTINE
720   !-------------------------------------------------------------------
730 Fft:              !
740   PRINT TABXY(25,10);" Calculating Fourier Transform -- Trace "&VAL$(Tr)
750   N3=2*N
760   J=1
770   FOR I=1 TO N3 STEP 2
780     IF I-J<0 THEN GOTO 800
790     IF I-J>=0 THEN GOTO 860
800     T=N1(J)
810     T1=N1(J+1)
820     N1(J)=N1(I)
830     N1(J+1)=N1(I+1)
840     N1(I)=T
850     N1(I+1)=T1
860     M=N3/2
870     IF J-M<=0 THEN GOTO 930
880     IF J-M>0 THEN GOTO 890
890     J=J-M
900     M=M/2
910     IF M-2<0 THEN GOTO 930
920     IF M-2>=0 THEN GOTO 870
930     J=J+M
940   NEXT I
950   M1=2
960   IF M1-N3<0 THEN GOTO 980
970   IF M1-N3>=0 THEN GOTO 1210
980   I1=2*M1
990   T2=2*PI/(S*M1)
1000  S1=SIN(T2/2)
1010  W1=-(2.*S1*S1)
1020  W2=SIN(T2)
1030  W3=1
1040  W4=0
1050  FOR M=1 TO M1 STEP 2
1060    FOR I=M TO N3 STEP I1
1070      J=I+M1
1080      T=W3*N1(J)-W4*N1(J+1)
1090      T1=W3*N1(J+1)+W4*N1(J)
1100      N1(J)=N1(I)-T
1110      N1(J+1)=N1(I+1)-T1
1120      N1(I)=N1(I)+T
1130      N1(I+1)=N1(I+1)+T1
1140    NEXT I
1150    T=W3
1160    W3=W3*W1-W4*W2+W3
1170    W4=W4*W1+T*W2+W4
```

```
1180  NEXT M
1190  M1=I1
1200  GOTO 960
1210  FOR I=1 TO N3 STEP 2
1220    IF S=-1 THEN N1(I)=N1(I)/N
1230    IF S=-1 THEN N1(I+1)=N1(I+1)/N
1240  NEXT I
1250  PRINT TABXY(25,10);"
1260  RETURN
1270  !---------------------------------------------------------------------
1280  ! DATAPLOT ROUTINE
1290  !---------------------------------------------------------------------
1300 Trace_plot:       !
1310  GRAPHICS ON
1320  VIEWPORT .05*X,.15*X,.30*Y,.95*Y
1330  WINDOW -42000,42000,-959,15
1340  MOVE N1(1),0
1350  FOR I=1 TO 959
1360    DRAW N1(2*I-1),-I
1370  NEXT I
1380  CLIP OFF
1390  LORG 4
1400  MOVE 0,2
1410  CSIZE 4
1420  LABEL VAL$(Tr)
1430  RETURN
1440  !---------------------------------------------------------------------
1450  !  AMPLITUDE SPECTRUM CALCULATION
1460  !---------------------------------------------------------------------
1470 Amplitude:          !
1480  PRINT TABXY(25,10);"Calculating amplitude spectrum  --  Trace ";Tr
1490  Ii=0.
1500  FOR I=1 TO N*2 STEP 2
1510    Ii=Ii+1
1520    P1(Ii)=SQR(N1(I)^2+N1(I+1)^2)
1530  NEXT I
1540  PRINT TABXY(25,10);"Normalizing amplitude spectrum   --  Trace ";Tr
1550  Amax=MAX(P1(*))
1560  Ii=0
1570  FOR I=1 TO N*2 STEP 2
1580    Ii=Ii+1
1590    P1(Ii)=P1(Ii)/Amax
1600  NEXT I
1610  PRINT TABXY(25,10);"
1620  RETURN
1630  !=====================================================================
1640 Spectrum_plot: !
1650  N=1024
1660  Sweep=1024/959*.192
1670  VIEWPORT .3*X,.9*X,.30*Y,.92*Y
1680  FRAME
1690  CLIP OFF
1700  LORG 5
1710  CSIZE 4
1720  WINDOW 0,N/16+1,0,1
1730  MOVE N/32,1.08
1740  LABEL "AMPLITUDE SPECTRUM"
1750  CLIP OFF
1760  MOVE N/32,-.15
1770  LABEL "FREQUENCY (Hz)"
```

```
1780  LORG 6
1790  CSIZE 3,1
1800  FOR F=0 TO 300 STEP 50
1810    MOVE F*Sweep,0
1820    DRAW F*Sweep,-.015
1830    MOVE F*Sweep,-.04
1840    LABEL USING "#,DDD";F
1850  NEXT F
1860  LORG 5
1870  CLIP ON
1880  MOVE 0,0
1890  FOR I=1 TO N/2+1
1900    Frek(I)=I/N
1910    DRAW I,P1(I)
1920  NEXT I
1930  RETURN
1940  END
```