WAVE THEORY FOR SEISMOGRAM SYNTHESIS

Chien-Ying Wang, B. S., M. S.

A Digest Presented to the Faculty of the Graduate
School of Saint Louis University in Partial
Fulfillment of the Requirements for the
Degree of Doctor of Philosophy

1981

DIGEST

A complete system of wave integral theory is established for purpose of synthesizing high quality and high frequency seismograms in plane layered media. The system consolidates the foundations of wave theory, and greatly facilitates its numerical application.

A classical contour integration method is extensively studied. Without introducing any sort of attenuation, the integration is taken directly along branch cuts and poles. An attempt to classify the constituents of seismograms from different integration contributions is discussed. Such a discussion proposes a new viewpoint for understanding wave fields.

The eigenfunctions of surface waves are found to have concise analytic solutions. These analytic forms not only provide a firm basis for theoretical development, but also provide a way to study high frequency signals and complicated structures.

The reflection and transmission properties of layer interfaces are reconsidered using a new approach. A simple method is proposed to decompose the wave fields, which can easily be incorporated into our system. Using this method, body as well as surface waves from a particular portion of structure are generated.

A new method for expressing seismic sources is explored, which enables us to isolate the fault orientation and receiver azimuthal dependences, thus facilitating the study of source mechanism. An inversion technique is developed to extract the instrument response coefficients. These coefficients were included in designing a recursive filter to describe the instrument effect.

Comparisons with other methods confirm that the new theory is both flexibile and reliable. The present study clarifies several ambiguities in the theory of the wave integral method and provides several new techniques for simulating wave propagation in the earth.

WAVE THEORY FOR SEISMOGRAM SYNTHESIS

Chien-Ying Wang, B. S., M. S.

A Dissertation Presented to the Faculty of the Graduate School of Saint Louis University in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

1981

COMMITTEE IN CHARGE OF CANDIDACY

Associate Professor Robert B. Herrmann

Chairman and Adviser

Professor Brian J. Mitchell

Professor Otto W. Nuttli

ACKNOWLEDGMENTS

Foremost, I wish to thank my advisor, Dr. Robert B. Herrmann, for providing continous guidance and stimulating ideas, throughout the duration of this work. I am especially grateful for his meticulous contributions to major parts of this dissertation.

I benefited greatly from discussions with Dr. Brian J. Mitchell and Dr. Otto W. Nuttli. Their interests provided much of the impetus and direction for the success of the work.

I want to thank Dr. David G. Harkrider for providing me the idea of using analytic solutions of eigenfunctions.

Special thanks are due to Eric J. Haug, Chandan K. Saikia, and David R. Russell. Their assistance in developing and maintaining the computer programs has been of great help. Finally, I thank my wife Tai-Chi and my son Albert, for their understanding and sacrifices in behalf of this effort.

This research was supported by National Science Foundation under Grant PFR-7909795.

TABLE OF CONTENTS

TABLE OF CONTENTS (CONT'D)

LIST OF TABLES

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (CONT'D)

LIST OF ILLUSTRATIONS (CONT'D)

LIST OF ILLUSTRATIONS (CONT'D)

LIST OF ILLUSTRATIONS (CONT'D)

LIST OF ILLUSTRATIONS (CONT'D)

CHAPTER I

INTRODUCTION

1.1 The Problem

In recent years synthetic seismograms have become increasingly useful as an aid to seismic data interpretation. Various techniques have been developed to calculate theoretical seismograms which are valid, at least, for plane-layered media. However, further improvements are still desirable. The problems of computational efficiency and accuracy, restricted by the use of a computer, continue to be a challenge. To solve them a thorough understanding of the nature of the problem and a detailed reexamination of the related theories are truly necessary.

The work of Thomson (1950) and Haskell (1953) first permitted the treatment of multi-layered media using matrix calculus. Prior to that time it was only possible to consider simple one- and two-layer models. Many developments based on the Thomson-Haskell technique have been pursued. The extension of their work to compound matrices and source specification are notable examples. Most of these developments, however, can be related to Haskell's work between 1953 and 1964. Because of simplicity in both concept and mathematics, Haskell's research provided an easily understandable approach to the problem. The purpose of this dissertation is to reconcile different formalisms and to establish a complete system to treat wave propagation in layered media using Haskell's work as a starting point. Such a system will provide a foundation for handling more complicated cases and for extending the theoretical studies.

The formulation of the method developed by Haskell is constructed in the frequency domain. The complete response at a particular frequency is represented by semi-infinite integrals with respect to wavenumber so as to automatically include all types of waves. To match the boundary conditions, the responses are described in terms of a layer matrix. There still exist several questions about the properties of this matrix and its compound form. A systematic approach to the problem will provide further insight to the Haskell matrices, and will promote their use.

After stacking the layer matrices over layers, time domain seismograms can be synthesized by performing any of several integrations in complex frequency-wavenumber plane. The behavior of the integrands must be known in order to choose the proper numerical method for integrations. This analysis also provides a framework for understanding the nature of different signals on the recorded seismogram, which in turn establishes a basis for exploring the effects of seismic source and earth structure on the seismograms.

The main application of the theory developed in this dissertation will be for synthesizing high quality seismograms at high frequencies. High frequency signals have become increasingly important in the studies of deep earth structure, earthquake source mechanism, and strong ground motion.

1.2 Historical Review

The era of the seismogram synthesis was opened by H. Lamb (1904), who generated the first synthetic seismogram for an impulse source acting upon a semi-infinite medium. Most early studies were for simple models such as liquid layers or one solid layer over a halfspace (see Ewing *et al*, 1957). With the advent of modern computers, the research rapidly grew to cover more complicated models and more sophisticated cases. In the last two decades, theoretical as well as numerical developments have progressed tremendously. We will now review the contribution of some papers on synthesizing seismogram. Most of them have formed the foundation of modern theoretical seismology.

In most theoretical developments, the evaluation of synthetic seismograms generally can be divided into two parts: first, the solution of an ordinary differential equation using transform methods with appropriate radiation and boundary conditions, and second, the evaluation of the corresponding inverse transforms. Several methods exist for both parts and many combinations are possible. One important approach, called 'generalized ray theory', uses the Laplace transform technique and generates waves at discrete time points by summing hundreds of rays. The basis of this method comes from Cagniard (1962) and de-Hoop (1960). Although this theory works well in predicting particular phases, it can be inaccurate and cumbersome when modeling long time duration seismograms for a multilayered structure. Hron(1972), Kennett (1974) and Wiggins and Madrid (1974) have made great efforts to improve the efficiency of calculating the responses of a large number of rays. A recent review of the method can be found in Pao and Gajewski (1977). Helmberger (1968) and Vered and Ben-Menahem (1976) present typical applications of the method to layered media.

Another approach, called 'wave theory', uses the Fourier transform technique and calculates all the waves excited in the structure by integrating over frequency and wavenumber (or slowness). This method encompasses normal-mode theory, especially for the determination of dispersion of surface waves (Press <u>et</u>

al, 1961). Because of increasing complexity at higher frequencies, the method suffers from computational inefficiency and stability problems. This dissertation will thoroughly explore this theory, especially at high frequencies.

In 1953 Haskell published a corrected version of Thomson's (1950) theory of elastic waves in a plane multilayered medium. Haskell's study introduced the 'matrix method' to seismic wave studies. This method provides a systematic approach and greatly facilitates numerical computation. Since then, the theory has been extended principally by Haskell (1963, 1964) and Harkrider (1964, 1970) to deal with the surface wave motion. Because of growing exponential terms in the matrix under certain conditions, the simple matrix method suffered from numerical problems which caused loss of precision. Knopoff (1964), Dunkin (1965), and Thrower (1965) reformulated the computational procedures using compound matrix extensions in which the minors of the layer matrices are propagated from interface to interface instead of the matrices themselves, so that the squared exponential terms never appear, thus controlling the precision problem. Randall (1967), Watson (1970), and Schwab and Knopoff (1970) made successful improvements in computational efficiency and accuracy. More recently Abo-Zena (1979) and Menke (1979) reexamined the matrix approach for extension to very high frequencies.

The existence of dispersive surface waves has been recognized since the early days of the science of seismology. Keilis-Borok (1960) presented a detailed study of surface waves generated in a layered medium. Vlaar (1966) applied eigenfunction theory in the analysis of Love wave generation. Saito (1967) developed a solution for surface wave excitation in terms of mutually orthogonal eigenfunctions for the generation of free oscillations in a radially inhomogeneous earth, and for surface waves in a vertically inhomogeneous flat earth. A recent contribution by Takeuchi and Saito (1972) summarized the previous studies, considering both theory and numerical methods. They applied the calculus of variations to derive the derivative-related quantities such as group velocity, attenuation factor, etc., from the surface wave eigenfunctions. This work provided a detailed mathematical basis for surface wave eigenfunction theory.

Hudson (1969a,b) extended the work of Haskell (1964) and Harkrider (1964) to synthesize seismic signals at teleseismic distances. Hudson's analysis is applicable for large epicentral distances, as all near-field terms are ignored. However, as shown by Herrmann (1978a), the truncation of these terms causes non-causal arrivals. Herrmann (1979) and Wang and

Herrmann (1980) expressed the Haskell formalism in a more concise form, and improved the integration method of Ewing _et al_ (1957) for synthesizing high quality seismograms. Other interesting methods to treat the problem of integration over the wavenumber-frequency domain appear in the work of Chapman (1978), Apsel (1979), and Bouchon (1979,1981).

Using another approach, Fuchs (1968) and Fuchs and Müller (1971) recognized that the reflectivity of some layers (reflection zone) can be isolated and excited by some incident waves from a 'transmission zone' to produce synthetic body wave seismograms. Their development was named the 'reflectivity method'. Fuchs (1971) simplified the solution by using the stationary phase approximation. The method is largely used in the study of crust or upper mantle structures from an artificial explosive source. However Kind and Müller (1975,1977) and Müller and Kind (1976) extended the method to include a double-couple point source, and after adding the earth flattening correction (Müller, 1971), they were able to simulate many real earth phases, such as Sn, ScS, SKS, etc. Kennett (1975) skillfully separated the transmission response beneath the source and the receiver in order to study the laterally varying structures. Another achievement of this method was the introduction of attenuation into the layers in the form of complex velocities (Kind, 1978). This modification enables a computation for only a short time window, even if the nonattenuated seismogram has a long duration.

Extending the systematic development by Gilbert and Backus (1966) of the 'propagator matrix', Kennett (1974) and his colleagues were able to express the matrix in terms of reflection and transmission properties of the stratified medium. This research bridged the gap between the wave approach and the ray method, and also provided an internal view of the layer matrix theory. Kennett et al (1978) extensively explored the symmetry properties of reflection and transmission coefficients. These symmetries not only reflect the theory of reciprocity, but provide a novel approach to exploit the properties of Haskell matrices. Kennett and Kerry (1979), Kennett (1980), and Kerry (1981) gave a complete derivation for this 'reflection and transmission coefficient' method and also proposed some interesting numerical evaluation techniques.

In searching for the roots of the period equation in the complex frequency plane, Gilbert (1964) described several wave-guide generated waves as leaky modes. Alsop (1970) interpreted these waves as a constructive interference of reverberating waves within a layer. Abramovici (1968) and Cochran et al (1970) found a relationship of this kind of mode to the regular normal mode. Watson (1972) gave a very detailed discussion of leaky modes by using real frequency-complex wavenumber analysis. In the observed data, it is believed that a dispersive body wave, called the PL wave, which arrives between the direct P and S waves, arises from the contribution of leaky modes. Oliver and Major (1960), Laster et al (1965), and Su and Dorman (1965) provided some real and experimental data analysis to reveal the properties of this wave. The success of improved computation with Haskell's method may lead to further insight into this phase.

CHAPTER II

WAVE INTEGRAL THEORY

The objective of this chapter is to establish a complete, self-contained, system for wave integral theory. The solution for the surface displacements is found using procedures parallel to those of Haskell (1964). The derivations are put forth in an easily understandable, step by step, way. Some interesting symmetry properties of the layer matrix or its compound form are revealed, which are then used to simplify the numerical application or theoretical extension.

A numerical integration technique of Herrmann (1978a, 1979) also is extended. The method requires contour integration in the complex wavenumber plane, the performance of which is complicated by the presence of singularities. The contour integration reduces to a consideration of pole residue contributions and branch line integrals. A detailed discussion of these aspects is undertaken, which is then used to improve the computational efficiency. The discussion also provides a framework for the investigation of different signals making up the complete seismogram.

2.1 Haskell's Matrix

In the present section a previous modification (Wang and Herrmann, 1980) of Haskell's theory (Haskell 1963, 1964) is further developed and revised. All of the derivations are made in a step-by-step manner for clarity. Several significant differences with respect to Haskell's papers are specifically noted. Such revisions are made not only to simplify the expressions, but also to compare them to other related formalisms to be discussed later. The three different types of waves, P, SV and SH, existing in the layered media are all included.

We define a semi-infinite elastic medium made up of N parallel, solid, homogeneous, isotropic layers (Figure 1). Each layer is characterized by the compressional wave velocity α , the shear wave velocity β , the density ρ , and the layer thickness d . Any linear variation of elastic properties can be approximated by many small layers (Fuchs, 1968). The m'th layer is bounded by the m and m+1 interfaces. Thus any quantities at the free surface are denoted by the subscript '1'. If a water layer is placed on the top, it will be assigned a layer index '0'. The parameters in the half-space are denoted by the subscript 'N'. In order to match the boundary conditions at the horizontal layer interfaces, a cylindrical coordinate

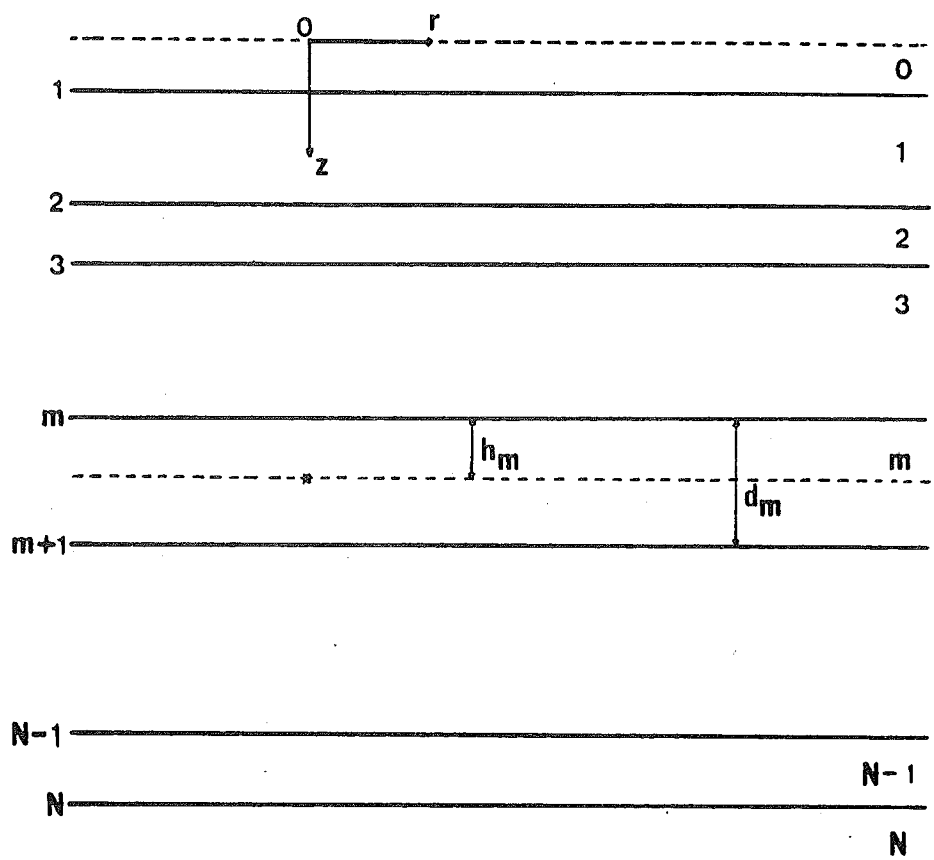


Figure 1. Direction of axes, numbering of layers and interfaces, and the depth of source in the source layer m.

system (r, ϑ , z) is chosen with origin on the free surface just above the source, and the z axis is taken positive downward.

In the following, expressions will be derived for the displacements from a point source using Haskell's notation (Haskell, 1964). Let us start from the very beginning. The displacement field can be expressed in terms of three different types of potentials, φ for P waves, ψ for SV waves, and χ for SH waves. The function φ is known as the scalar potential. ψ and χ are the vector potentials. Define

$$\mathbf{A} = \mathbf{e}_z \chi + \nabla \times (\mathbf{e}_z \psi),$$

where $\mathbf{e}_z$ is the unit vector in the z direction. Variables with bold cases represent vectors or matrices. The displacement $\mathbf{u}$ can be expressed as

$$\begin{aligned} \mathbf{u} &= \nabla\varphi + \nabla\times\mathbf{A} \\ &= \nabla\varphi + \nabla\times(\mathbf{e}_z\chi) + \nabla\times\nabla\times(\mathbf{e}_z\psi), \end{aligned} \qquad \text{(II-1-1)}$$

where the potential functions φ , ψ , and χ satisfy the wave equations:

$$\begin{aligned} \nabla^2\varphi &= \frac{1}{\alpha^2}\frac{\partial^2\varphi}{\partial t^2} \\ \nabla^2\psi &= \frac{1}{\beta^2}\frac{\partial^2\psi}{\partial t^2} \\ \nabla^2\chi &= \frac{1}{\beta^2}\frac{\partial^2\chi}{\partial t^2} \ . \end{aligned} \qquad \text{(II-1-2)}$$

Expanding the gradient and curl operators in equation (II-1-1) in cylindrical coordinates, we obtain the three components of displacement in terms of potentials:

$$u_r = \frac{\partial\varphi}{\partial r} + \frac{\partial\chi}{r\,\partial\vartheta} + \frac{\partial^2\psi}{\partial r\,\partial z}$$
$$u_z = \frac{\partial\varphi}{\partial z} + \frac{\partial^2\psi}{\partial z^2} - \nabla^2\psi \qquad \text{(II-1-3)}$$
$$u_\vartheta = \frac{\partial\varphi}{r\,\partial\vartheta} - \frac{\partial\chi}{\partial r} + \frac{\partial^2\psi}{r\,\partial\vartheta\,\partial z}\ .$$

If the time dependence is isolated by the Fourier transform factor $e^{i\omega t}$, the resulting Helmholtz equations (II-1-2) can be solved using characteristic functions of three cylindrical coordinates:

$$\varphi(r,\vartheta,z,\omega) = \left.\begin{matrix}\cos n\vartheta \\ \sin n\vartheta\end{matrix}\right\} \left.J_n(kr)\right\} \left.Z_1(z)\right\}$$
$$\psi(r,\vartheta,z,\omega) = \left.\begin{matrix}\cos n\vartheta \\ \sin n\vartheta\end{matrix}\right\} \left.J_n(kr)\right\} \left.Z_2(z)\right\} \qquad \text{(II-1-4)}$$
$$\chi(r,\vartheta,z,\omega) = \left.\begin{matrix}\cos n\vartheta \\ -\sin n\vartheta\end{matrix}\right\} \left.J_n(kr)\right\} \left.F_3(z)\right\},$$

where $J_n(kr)$ is the Bessel function of the first kind of order n. The solution involving the Bessel function of the second kind, Y_n, is not used since the solution must be valid at r = 0 where this function becomes unbounded. The subscript index, n, indicates azimuthal mode number, and k indicates the horizontal wavenumber. Z_1, Z_2, and F_3 are functions of z only, satisfying

$$\frac{d^2 Z_1}{dz^2} - \nu_\alpha^2 Z_1 = 0$$
$$\frac{d^2 Z_2}{dz^2} - \nu_\beta^2 Z_2 = 0 \qquad \text{(II-1-5)}$$
$$\frac{d^2 F_3}{dz^2} - \nu_\beta^2 F_3 = 0$$

where

$$\nu_\alpha = \begin{cases} \sqrt{k^2 - \dfrac{\omega^2}{\alpha^2}} & k \geqq \dfrac{\omega}{\alpha} \\ i\sqrt{\dfrac{\omega^2}{\alpha^2} - k^2} & k < \dfrac{\omega}{\alpha} \end{cases}$$

$$\nu_\beta = \begin{cases} \sqrt{k^2 - \dfrac{\omega^2}{\beta^2}} & k \geqq \dfrac{\omega}{\beta} \\ i\sqrt{\dfrac{\omega^2}{\beta^2} - k^2} & k < \dfrac{\omega}{\beta} \end{cases}$$

Now let us consider one homogeneous layer first. By substituting equation (II-1-4) into (II-1-3), we find

$$\begin{aligned} 4\pi\rho u_r(r,\vartheta,z,\omega) &= \cos n\vartheta\left[-\left\{-\left(\frac{dZ_2^c}{dz}+Z_1^c\right)k\right\}\frac{dJ_n(kr)}{dkr}-\left\{kF_3^c\right\}\frac{nJ_n(kr)}{kr}\right] \\ &\quad + \sin n\vartheta\left[\quad c\to s \quad\right] \\ 4\pi\rho u_z(r,\vartheta,z,\omega) &= \cos n\vartheta\left[\left\{k^2Z_2^c+\frac{dZ_1^c}{dz}\right\}J_n(kr)\right] \\ &\quad + \sin n\vartheta\left[\quad c\to s \quad\right] \\ 4\pi\rho u_\vartheta(r,\vartheta,z,\omega) &= \sin n\vartheta\left[\left\{kF_3^c\right\}\frac{dJ_n(kr)}{dkr}+\left\{-\left(\frac{dZ_2^c}{dz}+Z_1^c\right)k\right\}\frac{nJ_n(kr)}{kr}\right] \\ &\quad - \cos n\vartheta\left[\quad c\to s \quad\right], \end{aligned} \tag{II-1-6}$$

where $'c\to s'$ indicates the term in brackets above with c replaced by s. The constant ρ is included for simplifying the notation when the stresses are involved, and 4π for balancing the source terms which come from the Green's function. Using the transformed displacements given by equations (II-1-6), the transformed stresses across a horizontal plane are

$$4\pi T_{zr}(r,\vartheta,z,\omega) = 4\pi\mu\left\{\frac{\partial u_r}{\partial z}+\frac{\partial u_z}{\partial r}\right\}$$
$$= \cos n\vartheta\left[\omega^2\left\{[(\gamma-1)Z_2^c+\frac{\gamma}{k^2}\frac{dZ_1^c}{dz}]k\right\}\frac{dJ_n(kr)}{dkr}-\left\{\frac{\mu}{\rho}\frac{dF_3^c}{dz}k\right\}\frac{nJ_n(kr)}{kr}\right]$$
$$+\sin n\vartheta\left[\quad c\to s\quad\right]$$

$$4\pi T_{zz}(r,\vartheta,z,\omega) = 4\pi\left[\lambda\left(\frac{\partial u_r}{\partial r}+\frac{1}{r}\frac{\partial u_\vartheta}{\partial\vartheta}+\frac{u_r}{r}+\frac{\partial u_z}{\partial z}\right)+2\mu\frac{\partial u_z}{\partial z}\right]$$
$$= \cos n\vartheta\left[\omega^2\left\{(\gamma-1)Z_1^c+\gamma\frac{dZ_2^c}{dz}\right\}J_n(kr)\right] \qquad \text{(II-1-7)}$$
$$+\sin n\vartheta\left[\quad c\to s\quad\right]$$

$$4\pi T_{z\vartheta}(r,\vartheta,z,\omega) = 4\pi\mu\left[\frac{1}{r}\frac{\partial u_z}{\partial\vartheta}+\frac{\partial u_\vartheta}{\partial z}\right]$$
$$= \sin n\vartheta\left[\left\{\frac{\mu}{\rho}\frac{dF_3^c}{dz}k\right\}\frac{dJ_n(kr)}{dkr}+\omega^2\left\{[(\gamma-1)Z_2^c+\frac{\gamma}{k^2}\frac{dZ_1^c}{dz}]k\right\}\frac{nJ_n(kr)}{kr}\right]$$
$$-\cos n\vartheta\left[\quad c\to s\quad\right]$$

where

$$\gamma = \frac{2\beta^2k^2}{\omega^2} .$$

In equations (II-1-6) and (II-1-7), the azimuth dependent terms, the cosine and sine functions, are implied by the superscripts c and s, respectively. Since the model is plane layered, such a dependence arises solely from the orientation of the force system of the source. For homogeneous solutions, we first ignore the azimuthal terms and define the functions U_r, U_z, T_z, T_r, U_ϑ, and T_ϑ from the bracketed terms in equation (II-1-6) and (II-1-7) as follows

$$\begin{aligned}
\rho U_r &= -\left(\frac{dZ_2}{dz} + Z_1\right) k \\
\rho U_z &= k^2 Z_2 + \frac{dZ_1}{dz} \\
T_z &= (\gamma-1)Z_1 + \gamma\frac{dZ_2}{dz} \\
T_r &= \left[(\gamma-1)Z_2 + \frac{\gamma}{k^2}\frac{dZ_1}{dz}\right] k \\
\rho U_\vartheta &= F_3\, k \\
T_\vartheta &= \frac{\mu}{\rho}\frac{dF_3}{dz}\, k \quad .
\end{aligned} \tag{II-1-8}$$

The azimuthal terms dropped will be reconsidered when the source is introduced.

These functions are useful, since all of them satisfy the transformed boundary conditions: (a) the continuity of displacement and stress across the interfaces of layers, (b) the vanishing of stresses at the free surface, and (c) no upward waves in the bottom halfspace if the source is inside one of the upper layers. If we further require that waves decay exponentially in the halfspace, these functions are nothing but the eigenfunctions of a boundary value problem. In equation (II-1-8) there are extra k's compared to Haskell's (1964) definitions. The functions with an extra k are those which possess $\frac{dJ_n(kr)}{dkr}$ in the corresponding equations (II-1-6) and (II-1-7). Since $J_n(kr)$ or $J_{n-1}(kr)$ are characteristic functions in the r direction, these k's make the functions U_r, U_z, and

U_ϑ have dimensions of displacement. Also U_r differs by a minus sign and T_ϑ by ω^2 when compared to Haskell (1964) or Wang and Herrmann (1980). Such a particular definition will allow the layer matrix α and all other Haskell's matrices defined below to be more symmetric.

From equation (II-1-8), it is easy to find ordinary differential equations for these functions. Expressing equation (II-1-8) in matrix form, we have

$$\begin{bmatrix} U_r \\ U_z \\ T_z \\ T_r \\ U_\vartheta \\ T_\vartheta \end{bmatrix} = \begin{bmatrix} 0 & -\frac{k}{\rho} & -\frac{k}{\rho} & 0 & 0 & 0 \\ \frac{1}{\rho} & 0 & 0 & \frac{k^2}{\rho} & 0 & 0 \\ 0 & (\gamma-1) & \gamma & 0 & 0 & 0 \\ \frac{\gamma}{k} & 0 & 0 & (\gamma-1)k & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{k}{\rho} \\ 0 & 0 & 0 & 0 & \mu\frac{k}{\rho} & 0 \end{bmatrix} \begin{bmatrix} \frac{dZ_1}{dz} \\ Z_1 \\ \frac{dZ_2}{dz} \\ Z_2 \\ \frac{dF_3}{dz} \\ F_3 \end{bmatrix}. \qquad \text{(II-1-9)}$$

The inverse of this equation is

$$\begin{bmatrix} \frac{dZ_1}{dz} \\ Z_1 \\ \frac{dZ_2}{dz} \\ Z_2 \\ \frac{dF_3}{dz} \\ F_3 \end{bmatrix} = \begin{bmatrix} 0 & -\rho(\gamma-1) & 0 & k & 0 & 0 \\ -\rho\frac{\gamma}{k} & 0 & -1 & 0 & 0 & 0 \\ \rho\frac{(\gamma-1)}{k} & 0 & 1 & 0 & 0 & 0 \\ 0 & \rho\frac{\gamma}{k^2} & 0 & -\frac{1}{k} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{\rho}{\mu k} \\ 0 & 0 & 0 & 0 & \frac{\rho}{k} & 0 \end{bmatrix} \begin{bmatrix} U_r \\ U_z \\ T_z \\ T_r \\ U_\vartheta \\ T_\vartheta \end{bmatrix}. \qquad \text{(II-1-10)}$$

If we take the depth derivative on both sides of equation (II-1-9), replace the second z-derivatives of Z_1,

Z_2 , and F_3 by equation (II-1-5), combine with equation (II-1-10), and use $\rho\alpha^2 = \lambda + 2\mu$, $\rho\beta^2 = \mu$, we find that

$$\frac{d}{dz}\begin{bmatrix} U_r \\ U_z \\ T_z \\ T_r \\ U_\vartheta \\ T_\vartheta \end{bmatrix} = \begin{bmatrix} 0 & k & 0 & -\frac{\omega^2}{\mu} & 0 & 0 \\ -k\sigma & 0 & \omega^2\frac{\sigma}{\lambda} & 0 & 0 & 0 \\ 0 & -\rho & 0 & k & 0 & 0 \\ \rho-\xi\frac{k^2}{\omega^2} & 0 & -k\sigma & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{\mu} \\ 0 & 0 & 0 & 0 & \mu\nu_\beta^2 & 0 \end{bmatrix}\begin{bmatrix} U_r \\ U_z \\ T_z \\ T_r \\ U_\vartheta \\ T_\vartheta \end{bmatrix} \qquad \text{(II-1-11)}$$

where

$$\xi = \frac{4\mu(\lambda+\mu)}{\lambda+2\mu} \qquad \sigma = \frac{\lambda}{\lambda+2\mu} .$$

The matrix on the right hand side of this equation is denoted by **A** . Note that **A** is a skew-symmetric matrix, i.e., symmetric about the secondary diagonal axis. Such symmetric properties will reappear later. Actually, one part of the differential equations in (II-1-11) comes from the equation of motion and the other from the relationship between displacement and stress. The differential equation (II-1-11) forms the basis of the propagator matrix theory of Gilbert and Backus (1966) and the starting point for solving the eigenfunction problem by numerical integration (Takeuchi and Saito, 1972).

The z components of potentials satisfying equation (II-1-5) have the solutions

$$
\begin{aligned}
Z_1 &= A' e^{-\nu_\alpha z} + A'' e^{\nu_\alpha z} \\
Z_2 &= B' e^{-\nu_\beta z} + B'' e^{\nu_\beta z} \\
F_3 &= C' e^{-\nu_\beta z} + C'' e^{\nu_\beta z},
\end{aligned}
\qquad \text{(II-1-12)}
$$

where the single primes represent the waves propagating in the positive z direction, i.e., downward, and the double primes for waves in the negative direction, i.e., upward. Some authors normalize the coefficients in equation (II-1-12) by the energy flux in the z direction (Kennett et al, 1978), which is $\sqrt{2\nu_\alpha/\rho}$ for A and $\sqrt{2k^2\nu_\beta/\rho}$ for B and C. Normalization is not used here, but this point should be remembered in reference to the reflection and transmission coefficients. After substituting Z_1, Z_2, F_3 into equation (II-1-8), the U's and T's in the layer m can be expressed by the following matrix:

$$
\begin{bmatrix} U_r \\ U_z \\ T_z \\ T_r \\ U_\vartheta \\ T_\vartheta \end{bmatrix}_{m+1}
=
\begin{bmatrix}
-\frac{k}{\rho} & -\frac{\nu_\beta k}{\rho} & -\frac{k}{\rho} & \frac{\nu_\beta k}{\rho} & 0 & 0 \\
\frac{\nu_\alpha}{\rho} & \frac{k^2}{\rho} & -\frac{\nu_\alpha}{\rho} & \frac{k^2}{\rho} & 0 & 0 \\
(\gamma-1) & \gamma\nu_\beta & (\gamma-1) & -\gamma\nu_\beta & 0 & 0 \\
\frac{\gamma\nu_\alpha}{k} & (\gamma-1)k & -\frac{\gamma\nu_\alpha}{k} & (\gamma-1)k & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{k}{\rho} & \frac{k}{\rho} \\
0 & 0 & 0 & 0 & \frac{\mu k\nu_\beta}{\rho} & -\frac{\mu k\nu_\beta}{\rho}
\end{bmatrix}_m
diag
\begin{bmatrix} e^{\nu_\alpha z} \\ e^{\nu_\beta z} \\ e^{-\nu_\alpha z} \\ e^{-\nu_\beta z} \\ e^{\nu_\beta z} \\ e^{-\nu_\beta z} \end{bmatrix}_m
\begin{bmatrix} A'' \\ B'' \\ A' \\ B' \\ C'' \\ C' \end{bmatrix}_m
\qquad \text{(II-1-13)}
$$

where the subscript m is the layer index. We have denoted the quantities at the top of the m'th layer by m and at the bottom by m+1. From equation (II-1-11) and (II-1-13), it is obvious that the ϑ component can be separated from the r,z components. This represents the SH wave with particle motion parallel to the interfaces. For ease of expression, we will keep the three transformed components of motion together.

It is convenient to introduce the matrices

$$\mathbf{B} = [\, U_r, U_z, T_z, T_r, U_\vartheta, T_\vartheta \,]^T,$$

$$\mathbf{K} = [\, A'', B'', A', B', C'', C' \,]^T \ ,$$

$\mathbf{E}$ for the first matrix, and Λ for the diagonal second matrix on the right hand side of equation (II-1-13). Note that the matrix Λ describes phase variation along the depth direction. With these substitutions, equation (II-1-13) becomes

$$\mathbf{B}_{m+1} = \mathbf{E}_m \, \Lambda_m(z) \, \mathbf{K}_m \ . \qquad \text{(II-1-14)}$$

In equation (II-1-14) we have one of the most important equations of this dissertation. Matrix B , the eigenfunction vector is also called the motion-stress vector (Aki and Richards,1980) and has the convenient property of being continuous across interfaces except at the depth of the source. Matrix K is called the potential-constant vector which is composed of potential coefficients from equation (II-1-12). These coefficients are constant everywhere inside a homogeneous layer. In the reflectivity method, the vector plays an important role.

Let B_m be the value of B at the top of the m'th layer and B_{m+1} be its value at the bottom of this layer. Taking z = 0 in equation (II-1-14), we have

$$B_m = E_m\ K_m\ , \qquad \text{(II-1-15)}$$

and taking z = d_m , we have

$$B_{m+1} = E_m\ \Lambda_m(d_m)\ K_m\ . \qquad \text{(II-1-16)}$$

It is important to mention that although K_m is constant in the m'th layer, it can be premultiplied by the matrix Λ_m to become 'propagatable' when it is used to describe waves inside the layer. Therefore $\Lambda_m(0)\cdot K_m$ represents the potential at the top of layer m and $\Lambda_m(d_m)\cdot K_m$ is the potential at the bottom of this layer. By combining equations (II-1-15) and (II-1-16), we obtain a useful formula:

$$\begin{aligned} B_{m+1} &= E_m\ \Lambda_m\ (\ E_m^{-1}\ B_m\ ) \\ &= (\ E_m\ \Lambda_m\ E_m^{-1}\ )\ B_m\ = a_m\ B_m\ , \end{aligned} \qquad \text{(II-1-17)}$$

where we have defined

$$a_m = E_m\ \Lambda_m\ E_m^{-1}\ , \qquad \text{(II-1-18)}$$

and E_m^{-1} is

$$
E_m^{-1} = \frac{1}{2}\begin{bmatrix}
-\rho\frac{\gamma}{k} & -\rho\frac{(\gamma-1)}{\nu_\alpha} & -1 & \frac{k}{\nu_\alpha} & 0 & 0 \\
\rho\frac{(\gamma-1)}{\nu_\beta k} & \rho\frac{\gamma}{k^2} & \frac{1}{\nu_\beta} & -\frac{1}{k} & 0 & 0 \\
-\rho\frac{\gamma}{k} & \rho\frac{(\gamma-1)}{\nu_\alpha} & -1 & -\frac{k}{\nu_\alpha} & 0 & 0 \\
-\rho\frac{(\gamma-1)}{\nu_\beta k} & \rho\frac{\gamma}{k^2} & -\frac{1}{\nu_\beta} & -\frac{1}{k} & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{\rho}{k} & \frac{\rho}{\mu k \nu_\beta} \\
0 & 0 & 0 & 0 & \frac{\rho}{k} & -\frac{\rho}{\mu k \nu_\beta}
\end{bmatrix} . \qquad \text{(II-1-19)}
$$

Matrix α transfers the motion-stress vector through the layer; hence it is usually referred to as the layer matrix or matrix propagator. The elements of α are defined in Appendix A. This matrix possesses several interesting properties:

(1) $\alpha(z)$ is a function of z only, i.e., it propagates in the vertical direction, and this z-dependence arises solely from the diagonal matrix $\Lambda(z)$.

(2) $\alpha^{-1}(z) = \alpha(-z)$, since

$$
\begin{aligned}
\alpha^{-1}(z) &= [\, E\, \Lambda(z)\, E^{-1}\,]^{-1} \\
&= E\, \Lambda^{-1}(z)\, E^{-1} \\
&= E\, \Lambda(-z)\, E^{-1} \\
&= \alpha(-z) .
\end{aligned}
$$

(3) $a_{ij}^{-1} = (-1)^{i+j}\, a_{ij}$, this can be seen from Appendix A where terms with i+j = odd are related to $\sinh(\nu_{\alpha,\beta} z)$ which is an odd function, and terms with i+j = even are related to $\cosh(\nu_{\alpha,\beta} z)$ which is an even function.

(4) In a homogeneous layer,

$$\alpha_m(z_1+z_2) = \alpha_m(z_1)\,\alpha_m(z_2),$$

since the diagonal matrix Λ can be decomposed:

$$\begin{aligned}\alpha_m(z_1+z_2) &= \mathrm{E}_m\,\Lambda_m(z_1+z_2)\,\mathrm{E}_m^{-1}\\ &= \mathrm{E}_m\,\Lambda_m(z_1)\,\Lambda_m(z_2)\,\mathrm{E}_m^{-1}\\ &= \mathrm{E}_m\,\Lambda_m(z_1)\,\mathrm{E}_m^{-1}\,\mathrm{E}_m\,\Lambda_m(z_2)\,\mathrm{E}_m^{-1}\\ &= \alpha_m(z_1)\,\alpha_m(z_2).\end{aligned}$$

This is the property of a propagator matrix (Gilbert and Backus, 1966).

(5) All of the elements of matrix α are real for real k and ω, and most importantly the sinh functions always appear as $\sinh(\nu_{\alpha,\beta}z)/\nu_{\alpha,\beta}$ or $\sinh(\nu_{\alpha,\beta}z)\cdot\nu_{\alpha,\beta}$ which suppress the possible branch points due to the multivalued functions $\nu_{\alpha,\beta}$.

(6) α has a type of symmetry

$$\alpha_{ij} = \alpha_{5-j,5-i} \qquad \text{for } P-SV$$

$$\alpha_{55} = \alpha_{66} \qquad \text{for } SH \quad .$$

Properties (3) and (6) can be combined as

$$\alpha_{ij}^{-1} = (-1)^{i+j}\,\alpha_{5-j,5-i} \qquad \text{for } P-SV$$

$$\alpha_{ij}^{-1} = (-1)^{i+j}\,\alpha_{11-j,11-i} \qquad \text{for } SH \quad . \qquad \text{(II-1-20)}$$

In a similar way, the potential-constant vector can be transferred across the layer boundary by

$$K_{m+1} = E_{m+1}^{-1} E_m \Lambda_m(d_m) K_m \quad . \qquad \text{(II-1-21)}$$

$\Lambda_m(d_m) \cdot K_m$ is the potential located just above the m+1 interface and K_{m+1} just below this interface. Hence, $E_{m+1}^{-1} E_m$ contains the information about the reflection and transmission of waves passing through the m+1 interface. This property was extensively used by Kennett (1974) in his study of reflection and transmission coefficients.

In the above discussion, we constructed the displacement-stress field in a layered medium with the aid of the motion-stress vector B which in some sense represents the character of the boundaries. To connect these vectors between layers, a potential-constant vector K was defined in a more direct way than Haskell (1953,1964). Haskell (1964) defined $K = [A'+A'', A'-A'', B'-B'', B'+B'']^T$. This quantity is not convenient for application and was replaced by FK in Wang and Herrmann (1980). The potential-constant vector represents the waves inside the layers. With this new K we are able to find a diagonal matrix Λ , and consequently a better form for the layer matrix $\alpha = E\Lambda E^{-1}$. Matrix E consists of eigenvectors of matrix A in the differential equation (II-1-11), which will be further explored in section 2.3. It is noted that use of the matrix D of Haskell (1964) and matrix F of Wang and Herrmann (1980) has been avoided.

2.2 Compound Matrix

The compound matrix theory was first introduced to seismology by Dunkin (1965). This theory was proposed to treat the problem of loss of precision during calculation of the layer matrix in Haskell's (1953) original theory. Knopoff (1964) also developed a method using matrix representation to solve this problem. However, because of complexity in notation his method was not widely applied. The reason that Haskell's layer matrix needed to be extended to a compound matrix arises from the exponential terms contained in matrix Λ . During the calculation, these exponential terms, when they happen to be real functions as in the case of surface waves, will grow very large to obscure the significant figures of other factors. The remedy for this is to control the exponential values during the computation of these elements and/or to reformulate the matrix theory of Thomson (1950) or Haskell (1953) using compound matrix forms. In this section, we will discuss the compound matrix in detail and also consider the source representation to obtain complete solutions. The source function will be further explored in chapter V.

From Hudson (1969a) the source effect can be taken into account by means of a discontinuity of motion-stress vector across the source depth. Suppose that a source is in the m'th layer at a depth h_m beneath the m interface. The source vector S is defined as

$$S = B_m^+ - B_m^- \,, \tag{II-2-1}$$

where B_m^- , B_m^+ are the motion-stress vectors immediately above and below the source depth $z_m + h_m$, respectively. Such a source definition is different from that of Wang and Herrmann (1980), which is a modification of Haskell (1964), in that the S they used comes from the discontinuity of potential-constant vector K , i.e., $\Sigma = K_m^+ - K_m^-$. The relation of these two expression is just

$$S = E_m \Sigma \,. \tag{II-2-2}$$

Haskell's expression for the source is also useful when applied to other theories such as the reflectivity method (Fuchs, 1968). The introduction of revised source terms has the advantage that the factors ν_α and ν_β are no longer required.

It is straightforward to relate the motion-stress vector B_1 at the surface to B_m^- by the layer matrices in between:

$$B_m^- = a_m(h_m)\, a_{m-1} \cdots a_1 B_1 = ZB_1 \,. \tag{II-2-3}$$

Similarly, B_m^+ can be related to K_N in the half-space by

$$K_N = E_N^{-1} a_{N-1} \cdots a_m(d_m - h_m)\, B_m^+ = XB_m^+ \,. \tag{II-2-4}$$

Equations (II-2-1), (II-2-3), and (II-2-4) are further combined as

$$K_N = XB_m^+ = XS + XZB_1 = XS + RB_1 \qquad \text{(II-2-5)}$$

where

$$\begin{aligned} X &= E_N^{-1}\, \alpha_{N-1} \cdots \alpha_m(d_m - h_m) \\ Z &= \alpha_m(h_m) \cdots\cdots \alpha_1 \\ R &= X\,Z = E_N^{-1}\, \alpha_{N-1} \cdots \alpha_1 \;. \end{aligned} \qquad \text{(II-2-6)}$$

We have used property (4) of matrix α (page 24) to define matrix R.

Now we consider the boundary conditions at the free surface and the half-space. Vanishing of stresses at the free surface requires B_1 to be

$$B_1 = [\, U_{r_1}, U_{z_1}, 0, 0, U_{\vartheta_1}, 0\,]^T \;.$$

Inasmuch as the source is assumed to be in one of the layers, there can be no upwardly propagating waves in the half-space. Thus $A_N'' = B_N'' = C_N'' = 0$, and K_N takes the form

$$K_N = [\, 0, 0, A_N', B_N', 0, C_N'\,]^T.$$

The first two and the fifth rows in matrix equation (II-2-5) can be written as

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} X_{11} & X_{12} & X_{13} & X_{14} \\ X_{21} & X_{22} & X_{23} & X_{24} \end{bmatrix} \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \end{bmatrix} + \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix} \begin{bmatrix} U_{r_1} \\ U_{z_1} \end{bmatrix}$$

$$0 = [\, X_{55}\, ,\, X_{56}] \begin{bmatrix} S_5 \\ S_6 \end{bmatrix} + R_{55}\, U_{\vartheta_1}\ .$$

Here the P-SV and the SH components are separated. The free surface displacements are easily found to be

$$\begin{bmatrix} U_r \\ U_z \end{bmatrix}_1 = (-1) \begin{bmatrix} R_{22} & -R_{12} \\ -R_{21} & R_{11} \end{bmatrix} \begin{bmatrix} X_{1i}\, S_i \\ X_{2i}\, S_i \end{bmatrix} \Big/ R \,|^{12}_{12} \qquad i = 1, \ldots, 4$$

(II-2-7)

$$U_{\vartheta_1} = (-1)\,(\, X_{5j}\ S_j\,) \,/\, R_{55} \qquad j = 5, 6\ ,$$

where the use of summation convention for the subscripts and the compound matrix or second order subdeterminant definition $R\,|^{ij}_{kl} = R_{ik}R_{jl} - R_{il}R_{jk}$ are understood. This is the extent of Haskell's (1964) development. Let us consider the P-SV waves further. Since $\mathrm{R} = \mathrm{X\,Z}$, i.e., $R_{ij} = X_{ik} Z_{kj}$, equation (II-2-7) is further developed as

$$\begin{bmatrix} U_r \\ U_z \end{bmatrix}_1 = \begin{bmatrix} -S_i\, X\,|^{12}_{ij}\, Z_{j2} \\ S_i\, X\,|^{12}_{ij}\, Z_{j1} \end{bmatrix} \Big/ R\,|^{12}_{12}\ . \qquad \text{(II-2-8)}$$

A compound matrix can be expressed as the matrix product of its compound sub-matrices (Dunkin, 1965). Hence from equation (II-2-6)

$$X\,|^{12}_{ij} = E_N^{-1}|^{12}_{mn}\ a_{N-1}|^{mn}_{op} \cdots\ a_{m+1}|^{qr}_{st}\ a_m(d_m - h_m)\,|^{st}_{ij}$$

(II-2-9)

$$R\,|^{12}_{12} = E_N^{-1}|^{12}_{mn}\ \cdots\ a_2|^{qr}_{st}\ a_1|^{st}_{12}\ .$$

The elements of compound matrices $E_N^{-1}|_{ij}^{12}$ and $\alpha|_{kl}^{ij}$ are listed in Appendix B. It is important to point out that in deriving $\alpha|_{kl}^{ij}$ from matrix α_{ij} we are able to remove the questionable square exponential terms by the analytic equality $\cosh^2(\nu z) - \sinh^2(\nu z) = 1$. This is the basic reason that the compound matrix is better than the simple matrix in Haskell's theory. In Appendix B, we have factored out the quantity $1/(4k\nu_{\alpha_N}\nu_{\beta_N})$ when defining $E_N^{-1}|_{ij}^{12}$ to remove the possible singularities from $\nu_{\alpha N}$ and $\nu_{\beta N}$. This quantity is usually canceled out by division of similar terms in the denominator of equation (II-2-8). Therefore it will be suppressed in our calculation. However, in other cases, this term might not be discarded so easily, such as when considering the stresses (Baumgardt, 1980). This point should be noted when the compound matrix $E_N^{-1}|_{ij}^{12}$ is included in the formulation.

The compound matrix $\alpha|_{kl}^{ij}$ can be looked at as a 6 by 6 matrix, if we assign the indices as

$$12 \to 1 \quad 13 \to 2 \quad 14 \to 3$$
$$23 \to 4 \quad 24 \to 5 \quad 34 \to 6 .$$

In Appendix B, we find that the third and fourth rows or columns of compound matrix $\alpha|_{kl}^{ij}$ are equivalent in a particular way. This can be explained mathematically in Appendix C by means of two properties of compound matrices. As indicated by Watson (1970), this equivalence permits us to reduce the 6 by 6 matrix to a

5 by 5 matrix by discarding the third column and row and replacing the fourth row by [$2\alpha|^{23}_{12}$, $2\alpha|^{23}_{13}$, $2\alpha|^{23}_{23} - 1$, $2\alpha|^{23}_{24}$, $2\alpha|^{23}_{34}$] during the matrix multiplication. The $E_N^{-1}|^{12}_{ij}$ matrix also drops the third component $E_N^{-1}|^{12}_{14}$. Such a reduction saves some storage space and computer time. The third component dropped can be retrieved easily, since it is exactly the same as the old fourth component.

Because of the symmetry exhibited by matrices α and E^{-1} , there exist some interesting properties of their compound forms. As shown in Appendix C and equation (II-1-2Ø), the compound matrices X and R , which consist of E^{-1} and α , possess the following properties:

$$
\begin{aligned}
Y^{-1}|^{ij}_{34} &= (-1)^{i+j+1}\, q_N\, Y|^{12}_{5-j,5-i} \\
Y\,|^{12}_{14} &= Y|^{12}_{23} \\
Y^{-1}|^{14}_{34} &= Y^{-1}|^{23}_{34}
\end{aligned}
\qquad \text{(II-2-10)}
$$

where Y might be E^{-1} , X , or R , and the constant $q_N = \dfrac{4k^2 \nu_{\alpha N} \nu_{\beta N}}{\rho_N^2}$. This constant consists of the normalization factors we have mentioned in equation (II-1-12). Matrix Z , which consists of α's , also has the properties:

$$
\begin{aligned}
Z_{ij}^{-1} &= (-1)^{i+j}\, Z_{5-j,5-i} \\
Z^{-1}|^{ij}_{kl} &= (-1)^{i+j+k+l}\, Z|^{5-l,5-k}_{5-j,5-i} \; .
\end{aligned}
$$

These symmetry properties are a characteristic of

Haskell's formulation, although they arise from a somewhat arbitrary choice of motion-stress and potential-constant vectors in the last section. In Haskell's work (1964) or other similar formulations, these properties are more or less missing. These properties will be of particular interest when relating the eigenfunction theory and matrix theory in the next chapter.

The surface displacements in the forms of equation (II-2-8) are stable for calculation, since they include $X|^{12}_{kl}$ which imposes the boundary conditions from halfspaces all the time. However, Harkrider (1964) and Harvey (1981) used another form. Since $R = X Z$ or equivalently $X = R Z^{-1}$ we have

$$\begin{aligned} X|^{12}_{ij} Z_{jp} &= (X_{1i} X_{2j} - X_{1j} X_{2i}) Z_{jp} \\ &= R_{1k} Z^{-1}_{ki} R_{2l} (Z^{-1}_{lj} Z_{jp}) - R_{1m} (Z^{-1}_{mj} Z_{jp}) R_{2k} Z^{-1}_{ki} \\ &= R_{1k} Z^{-1}_{ki} R_{2p} - R_{1p} R_{2k} Z^{-1}_{ki} \\ &= R|^{12}_{kp} Z^{-1}_{ki} \; . \end{aligned}$$

Hence

$$\begin{bmatrix} U_r \\ U_z \end{bmatrix}_1 = (-1) \begin{bmatrix} R|^{12}_{k2} Z^{-1}_{ki} S_i \\ -R|^{12}_{k1} Z^{-1}_{ki} S_i \end{bmatrix} \Big/ R|^{12}_{12} \; . \qquad \text{(II-2-11)}$$

Numerically this form is not as good as equation (II-2-8), since it requires more calculations with $R|^{12}_{ij}$ than with $X|^{12}_{ij}$. However it is a useful expression for deriving other properties.

At this point, it is interesting to consider the dispersion property of surface waves. It is known that the dispersion is determined by the velocity structure and is independent of sources. Assuming $S = 0$ and $K_N = R\,B_1$, we obtain

$$\begin{bmatrix} 0 \\ 0 \\ A' \\ B' \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & \cdot & \cdot \\ R_{21} & R_{22} & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{bmatrix} \begin{bmatrix} U_{r_1} \\ U_{z_1} \\ 0 \\ 0 \end{bmatrix} .$$

The first two rows give

$$\frac{U_{r_1}}{U_{z_1}} = -\frac{R_{12}}{R_{11}} = -\frac{R_{22}}{R_{21}}$$

and the last equality defines the period equation $R\,|^{12}_{12} = 0$. U_{r_1}/U_{z_1} is called the ellipticity of Rayleigh waves at the free surface, which is usually denoted by ε . By applying the arithmetic equality

$$\frac{b}{a} = \frac{d}{c} = \frac{bx - dy}{ax - cy}$$

where x and y are suitably arbitrary numbers, it is not difficult to find that

$$\varepsilon \equiv \frac{U_{r_1}}{U_{z_1}} = -\frac{R\,|^{12}_{23}}{R\,|^{12}_{13}} = -\frac{R\,|^{12}_{24}}{R\,|^{12}_{14}} \qquad \text{(II-2-12)}$$

(This is the derivation of Equation 20 in Harkrider (1970)). Furthermore, another expression exists for the ellipticity in terms of R^{-1}. By using the symmetry properties of compound **R** (equation (II-2-10)), it is found that

$$\mathcal{E} = \frac{R^{-1}|^{13}_{34}}{R^{-1}|^{23}_{34}} = \frac{R^{-1}|^{14}_{34}}{R^{-1}|^{24}_{34}} \,.$$

The period equation in this case is $R^{-1}|^{34}_{34} = 0$.

The solutions in equation (II-2-8) include the source terms S_i. For the surface waves in which $R\,|^{12}_{12} = 0$, the ellipticity of Rayleigh waves is known to be independent of the source. This can be shown by another form of the solution in equation (II-2-11):

$$\mathcal{E} \equiv \frac{U_{r_1}}{U_{z_1}} = -\,\frac{R\,|^{12}_{k2}\, Z^{-1}_{ki}\, S_i}{R\,|^{12}_{k1}\, Z^{-1}_{ki}\, S_i}$$

$$= -\,\frac{R\,|^{12}_{23}\,\left(Z^{-1}_{3i}\, S_i + \dfrac{R\,|^{12}_{24}}{R\,|^{12}_{23}}\, Z^{-1}_{4i}\, S_i \right)}{R\,|^{12}_{13}\,\left(Z^{-1}_{3i}\, S_i + \dfrac{R\,|^{12}_{14}}{R\,|^{12}_{13}}\, Z^{-1}_{4i}\, S_i \right)} \,. \qquad \text{(II-2-13)}$$

Using an equality for any compound matrix $R\,|^{12}_{ij}$ (Abramovici, 1968),

$$R\,|^{12}_{12}\; R\,|^{12}_{34} + R\,|^{12}_{14}\; R\,|^{12}_{23} = R\,|^{12}_{13}\, R\,|^{12}_{24} \,,$$

however since $R\,|^{12}_{12} = 0$, equation (II-2-13) becomes

$$\mathcal{E} = -\,\frac{R\,|^{12}_{23}}{R\,|^{12}_{13}} \,.$$

Thus we have shown that the ellipticity is really a quantity determined from the layer response only and is independent of the source.

The solutions for the Fourier transformed displacements as expressed in equation (II-1-6), after summing the mode numbers over horizontal and azimuthal directions, have the forms

$$u_z(r,\vartheta,0,\omega) = \sum_{n=0}^{\infty} \int_0^{\infty} dk \ \{ (U_z^{nc} \cos n\vartheta + U_z^{ns} \sin n\vartheta) J_n(kr) / F_R \}$$

$$\begin{aligned} -u_r(r,\vartheta,0,\omega) = \sum_{n=0}^{\infty} \int_0^{\infty} dk \ \{ & (U_r^{nc} \cos n\vartheta + U_r^{ns} \sin n\vartheta) J_{n-1}(kr) / F_R \\ & - (\frac{n}{kr}) (U_r^{nc} \cos n\vartheta + U_r^{ns} \sin n\vartheta) J_n(kr) / F_R \\ & + (\frac{n}{kr}) (U_r^{ns} \cos n\vartheta - U_\vartheta^{nc} \sin n\vartheta) J_n(kr) / F_L \} \end{aligned} \qquad \text{(II-2-14)}$$

$$\begin{aligned} -u_\vartheta(r,\vartheta,0,\omega) = \sum_{n=0}^{\infty} \int_0^{\infty} dk \ \{ & (U_\vartheta^{ns} \cos n\vartheta - U_\vartheta^{nc} \sin n\vartheta) J_{n-1}(kr) / F_L \\ & - (\frac{n}{kr}) (U_\vartheta^{ns} \cos n\vartheta - U_\vartheta^{nc} \sin n\vartheta) J_n(kr) / F_L \\ & + (\frac{n}{kr}) (U_r^{ns} \cos n\vartheta - U_\vartheta^{nc} \sin n\vartheta) J_n(kr) / F_R \} \ , \end{aligned}$$

where $U_z^{nc,s}$, $U_r^{nc,s}$, $U_\vartheta^{nc,s}$, F_R and F_L are

$$\begin{aligned} U_z^{nc,s} &= S_i^{nc,s} X |_{ij}^{12} \ Z_{j1} \\ U_r^{nc,s} &= - S_i^{nc,s} X |_{ij}^{12} \ Z_{j2} \\ U_\vartheta^{nc,s} &= - X_{5j} \ S_j^{nc,s} \\ F_R &= R |_{12}^{12} \\ F_L &= R_{55} \ . \end{aligned}$$

There is now no doubt that an earthquake can be represented by a double-couple source without moment model (Aki and Richards, 1980, p.43). Haskell (1963) used **n** for the vector normal to the fault and **f** for the direction of force to describe the source. For practical applications, the dip d , strike φ of the fault plane and the slip s of movement on this plane are usually preferred. The relation of these quantities to **n** and **f** are just

$$n_1 = -\sin d \sin\varphi \qquad f_1 = \cos s \cos\varphi + \sin s \cos d \sin\varphi$$
$$n_2 = \sin d \cos\varphi \qquad f_2 = \cos s \sin\varphi - \sin s \cos d \cos\varphi$$
$$n_3 = -\cos d \qquad f_3 = -\sin s \sin d \;.$$

The Fourier transformed displacements generated by such a dislocation source, for which n = Ø,1,2 in equation (II-2-14), have the forms:

$$\begin{aligned} u_z(r,\vartheta,0,\omega) &= ZSS \cdot R_{ss} + ZDS \cdot R_{ds} + ZDD \cdot R_{dd} \\ u_r(r,\vartheta,0,\omega) &= RSS \cdot R_{ss} + RDS \cdot R_{ds} + RDD \cdot R_{dd} \\ u_\vartheta(r,\vartheta,0,\omega) &= TSS \cdot R'_{ss} + TDS \cdot R'_{ds} \end{aligned} \qquad \text{(II-2-15)}$$

where

$$\begin{aligned} R_{ss} &= (f_1 n_1 - f_2 n_2)\cdot\cos 2\vartheta + (f_1 n_2 + f_2 n_1)\cdot\sin 2\vartheta \\ &= \sin d \cos s \sin 2(\vartheta-\varphi) + \frac{1}{2}\sin 2d \sin s \cos 2(\vartheta-\varphi) \\ R_{ds} &= (f_1 n_3 + f_3 n_1)\cdot\cos\vartheta + (f_2 n_3 + f_3 n_2)\cdot\sin\vartheta \\ &= \cos 2d \sin s \sin(\vartheta-\varphi) - \cos d \cos s \cos(\vartheta-\varphi) \\ R_{dd} &= f_3 n_3 \\ &= \frac{1}{2}\sin s \sin 2d \\ R'_{ss} &= (f_1 n_2 + f_2 n_1)\cdot\cos 2\vartheta - (f_1 n_1 - f_2 n_2)\cdot\sin 2\vartheta \\ &= \sin d \cos s \cos 2(\vartheta-\varphi) - \frac{1}{2}\sin 2d \sin s \sin 2(\vartheta-\varphi) \\ R'_{ds} &= (f_2 n_3 + f_3 n_2)\cdot\cos\vartheta - (f_1 n_3 + f_3 n_1)\cdot\sin\vartheta \\ &= \cos d \cos s \sin(\vartheta-\varphi) + \cos 2d \sin s \cos(\vartheta-\varphi) \end{aligned}$$

$$ZSS = \int_0^\infty (1/F_R)\,\{\, S_i^2\, X\,|_{ij}^{12}\, Z_{j1} \,\}\, J_2(kr)\, dk$$

$$ZDS = \int_0^\infty (1/F_R)\,\{\, S_i^1\, X\,|_{ij}^{12}\, Z_{j1} \,\}\, J_1(kr)\, dk$$

$$ZDD = \int_0^\infty (1/F_R)\,\{\, S_i^0\, X\,|_{ij}^{12}\, Z_{j1} \,\}\, J_0(kr)\, dk$$

$$RSS = \int_0^\infty (1/F_R)\,\{\, S_i^2\, X\,|_{ij}^{12}\, Z_{j2} \,\}\, J_1(kr)\, dk$$

$$- \int_0^\infty (1/F_R) \{ S_i^2 X|_{ij}^{12} Z_{j2} \} 2J_2(kr)/kr \; dk$$

$$+ \int_0^\infty (1/F_L) \{ S_j^6 X_{5j} \} 2J_2(kr)/kr \; dk$$

$$RDS = \int_0^\infty (1/F_R) \{ S_i^1 X|_{ij}^{12} Z_{j2} \} J_0(kr) \; dk$$

$$- \int_0^\infty (1/F_R) \{ S_i^1 X|_{ij}^{12} Z_{j2} \} J_1(kr)/kr \; dk$$

$$+ \int_0^\infty (1/F_L) \{ S_j^5 X_{5j} \} J_1(kr)/kr \; dk$$

$$RDD = - \int_0^\infty (1/F_R) \{ S_i^0 X|_{ij}^{12} Z_{j2} \} J_1(kr) \; dk$$

$$TSS = \int_0^\infty (1/F_L) \{ X_{5j} \; S_j^6 \} J_1(kr) \; dk$$

$$- \int_0^\infty (1/F_L) \{ X_{5j} \; S_j^6 \} 2J_2(kr)/kr \; dk$$

$$+ \int_0^\infty (1/F_R) \{ S_i^2 X|_{ij}^{12} Z_{j2} \} 2J_2(kr)/kr \; dk$$

$$TDS = \int_0^\infty (1/F_L) \{ X_{5j} \; S_j^5 \} J_0(kr) \; dk$$

$$- \int_0^\infty (1/F_L) \{ X_{5j} \; S_j^5 \} J_1(kr)/kr \; dk$$

$$+ \int_0^\infty (1/F_R) \{ S_i^1 X|_{ij}^{12} Z_{j2} \} J_1(kr)/kr \; dk \; ,$$

with

$$\begin{aligned} &4\pi\rho_m \omega^2 S_2^0 = 4k \; k_{\alpha_m}^2 && 4\pi\omega^2 S_4^0 = 2k^2 [(2\beta_m/\alpha_m)^2 - 3] \\ &4\pi\rho_m \omega^2 S_1^1 = -2k \; k_{\beta_m}^2 && 4\pi\omega^2 S_4^2 = -2k^2 \qquad \text{(II-2-16)} \\ &4\pi\rho_m \beta_m^2 S_5^1 = -2k && 4\pi S_6^2 = 2k^2 \; . \end{aligned}$$

All other S_i^n equal zero. The solutions and source forms for other types of sources will be given in chapter V.

In the above equations, SS represents a strike-slip type of source, DS represents a dip-slip type of source, and DD represents a 45° dip-slip type of source. It is apparent that these three components are all that are necessary to represent the P-SV motion for any shear-dislocation source (Harkrider, 1976). The SH motion only needs SS and DS components. RSS, RDS, TSS, and TDS in equation (II-2-15) also include the near-field terms. These terms decrease faster with r than the others, and thus, are important only at short distances or low frequencies. The near-field terms in the r and t directions have exactly the same forms. It is worthwhile to indicate that the near-field terms have a higher order Bessel function than their far-field counterparts. The calculation of near-field terms requires special care. This will be studied in section 2.4.

2.3 Comparison with Other Formalisms

Many approaches have been developed to treat wave propagation in isotropic and homogeneous layered media. In this section, we will discuss three typical approaches from the standpoint of the system constructed in this dissertation. It will be shown that our system is related to these other approaches with only a little modification. Examples describing the merit of our approach will be given later.

The Eigenvector of ODE

We have already mentioned that the eigenfunction theory plays an important role in establishing our matrix system. The mathematical foundation of matrix theory was provided by Gilbert and Backus (1966) using the 'propagator matrix' description. The mathematical connection is enlarged in the following discussion.

The motion-stress vector satisfies the first order differential equation (II-1-11):

$$\frac{d}{dz}\mathbf{B} = \mathbf{A}\,\mathbf{B}\,, \tag{II-3-1}$$

where $\mathbf{A}$ is a matrix representing the material property. Within a given layer $\mathbf{A}$ is a constant, and the solution takes the form

$$\mathbf{B}(z) = e^{(z-z_0)\cdot\mathbf{A}}\,\mathbf{B}(z_0)\,,$$

where z_0 is the reference depth. The function $e^{(z-z_0)\cdot\mathbf{A}}$, called the matricant or layer matrix for a homogeneous layer, is defined by Sylvester's theorem (Hildebrand, 1965, p.61). Compared to equation (II-1-17), the layer matrix defined in equation (II-1-18) must be

$$a = e^{(z-z_0)\cdot\mathbf{A}}\,. \tag{II-3-2}$$

Applying matrix multiplication, with $\mathbf{E}^{-1}$ and $\mathbf{E}$ from equations (II-1-13) and (II-1-19), equation (II-3-2) becomes

$$E^{-1}\alpha E = E^{-1} e^{(z-z_0)\cdot A} E = e^{(z-z_0) E^{-1}AE} .$$

Since $\Lambda(z) = E^{-1}\alpha E$ (equation II-1-18), we obtain an interesting form:

$$E^{-1} A E = \begin{bmatrix} \nu_\alpha & 0 & 0 & 0 & 0 & 0 \\ 0 & \nu_\beta & 0 & 0 & 0 & 0 \\ 0 & 0 & -\nu_\alpha & 0 & 0 & 0 \\ 0 & 0 & 0 & -\nu_\beta & 0 & 0 \\ 0 & 0 & 0 & 0 & \nu_\beta & 0 \\ 0 & 0 & 0 & 0 & 0 & -\nu_\beta \end{bmatrix} \equiv V .$$

It is found that matrix A is diagonalized in a way similar to that of the characteristic-value problem. According to eigenvalue theory, E must be constructed from the eigenvectors of A and $\pm\nu_\alpha$, $\pm\nu_\beta$ are its eigenvalues. This fact is confirmed as we actually obtain the eigenvalues from matrix A (Aki and Richards, 1980, p.275). Therefore the eigenvectors e_k's of A constitute the successive columns of the E matrix:

$$E = \begin{bmatrix} | & | & | & | & & \\ e_1 & e_2 & e_3 & e_4 & & \\ | & | & | & | & & \\ & & & & | & | \\ & & & & e_5 & e_6 \\ & & & & | & | \end{bmatrix} .$$

The corresponding eigenvalues λ's are $\nu_\alpha, \nu_\beta, -\nu_\alpha, -\nu_\beta, \nu_\beta, -\nu_\beta$. The e_k's are mutually orthogonal.

With this view of the E matrix, matrix B in equation (II-3-1) can also be expressed in terms of eigenvectors and eigenvalues. Project vector B into the direction of e_1 eigenvector, and denote this component by M_1 :

$$\mathbf{M}_1 = \mathbf{B}\cdot\mathbf{e}_1 = f_1(z)\cdot\mathbf{e}_1 \ .$$

Because of the linear independence of e_k's, M_1 still satisfies the differential equation (II-3-1):

$$\frac{d}{dz}\left(f_1(z)\,\mathbf{e}_1\right) = \mathbf{A}\cdot\left(f_1(z)\,\mathbf{e}_1\right)$$
$$\left(\frac{d}{dz} f_1(z)\right)\mathbf{e}_1 = (\mathbf{A}\cdot\mathbf{e}_1)\, f_1(z) = (\lambda_1\mathbf{e}_1)\, f_1(z) \ ,$$

i.e.,

$$\frac{d}{dz} f_1(z) = \lambda_1 f_1(z) \ .$$

Hence in the e_1 direction, we have the solution

$$\mathbf{M}_1 = e^{\lambda_1(z-z_0)}\mathbf{e}_1 \ .$$

Repeating the same procedure for the other components, we obtain the solution to the ordinary differential equation (II-3-1):

$$\mathbf{M} = \begin{bmatrix} | & | & | & | & & \\ \mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3 & \mathbf{e}_4 & & \\ | & | & | & | & & \\ & & & & | & | \\ & & & & \mathbf{e}_5 & \mathbf{e}_6 \\ & & & & | & | \end{bmatrix} \begin{bmatrix} e^{\lambda_1\cdot(z-z_0)} & & & \\ & e^{\lambda_2\cdot(z-z_0)} & & 0 \\ & & \ddots & \\ & 0 & & \ddots \end{bmatrix} = \mathbf{E}\cdot\Lambda \ .$$

This form of solution is called a fundamental matrix.

The propagator matrix is defined from it (Gilbert and Backus, 1966, eq.2.4):

$$\begin{aligned} \mathbf{\alpha}(z-z_0) &= \mathbf{M}(z)\,\mathbf{M}^{-1}(z_0) = \mathbf{E}\,\Lambda(z)\,[\mathbf{E}\,\Lambda(z_0)]^{-1} \\ &= \mathbf{E}\,\Lambda(z)\,\Lambda(-z_0)\,\mathbf{E}^{-1} \\ &= \mathbf{E}\,\Lambda(z-z_0)\,\mathbf{E}^{-1}\,. \end{aligned}$$

The most general solution of equation (II-3-1) is described by a linear combination of the fundamental matrix $\mathbf{M}$,

$$\mathbf{B} = \mathbf{M}\,\mathbf{K} = \mathbf{E}\,\Lambda\,\mathbf{K}$$

where $\mathbf{K}$ is any constant vector. It follows that the differential equation (II-3-1) can be written in the form:

$$\begin{aligned} \frac{d}{dz}(\mathbf{E}\,\Lambda(z)\,\mathbf{K}) &= \mathbf{A}\,(\mathbf{E}\,\Lambda(z)\,\mathbf{K}) \\ \frac{d}{dz}(\Lambda(z)\,\mathbf{K}) &= (\mathbf{E}^{-1}\mathbf{A}\mathbf{E}\,)\,\Lambda(z)\,\mathbf{K} \\ &= \mathbf{V}\,(\Lambda(z)\,\mathbf{K})\,, \end{aligned} \qquad \text{(II-3-3)}$$

where we have used the fact that $\mathbf{E}$ and $\mathbf{K}$ are independent of z. Since $\mathbf{V}$ is a diagonal matrix, it is said that the wave field has been 'decoupled' (Claerbout, 1976, p.169). It is easy to see that vector $\mathbf{K}$ is just the potential-constant vector defined before, which consists of upgoing and downgoing waves. These waves flow up and down in homogeneous regions without interacting with each other.

Compound Matrix for High Frequencies

The compound matrix $X|_{ij}^{12}$ expressed in equation (II-2-9) appears as a (1X6) matrix during compound-compound multiplication (Dunkin, 1965):

$$X|_{ij}^{12} = [\, X|_{12}^{12} \,,\, X|_{13}^{12} \,,\, X|_{14}^{12} \,,\, X|_{23}^{12} \,,\, X|_{24}^{12} \,,\, X|_{34}^{12} \,]^T \,.$$

When $X|_{ij}^{12}$ is multiplied by a regular matrix as in equation (II-2-8), it can be viewed as a (4X4) anti-symmetric matrix:

$$X|_{ij}^{12} = \begin{bmatrix} 0 & X|_{12}^{12} & X|_{13}^{12} & X|_{14}^{12} \\ -X|_{12}^{12} & 0 & X|_{23}^{12} & X|_{24}^{12} \\ -X|_{13}^{12} & -X|_{23}^{12} & 0 & X|_{34}^{12} \\ -X|_{14}^{12} & -X|_{24}^{12} & -X|_{34}^{12} & 0 \end{bmatrix} .$$

This property provides an alternative method of formulation.

In recent papers, Abo-Zena (1979) and Menke (1979) decomposed the layer matrix multiplication into a form with recursive relations:

$$Y_m = A_m^T \cdots A_{N-1}^T \,\{\, [EA]^T \,[EB] - [EB]^T \,[EA] \,\}\, A_{N-1} \cdots A_m \qquad \text{(II-3-4)}$$

(Abo-Zena, equation 40), where $\{[EA]^T[EB] - [EB]^T[EA]\}$ is an anti-symmetric matrix, which is actually equivalent to our $E^{-1}|_{kl}^{12}$. After a tedious expansion, we find that

$$Y|_{ij}^{12} A|_{kl}^{ij} = A_{ki}^{T}\, Y|_{ij}^{12} A_{jl} \quad .$$

The recursive procedure (II-3-4) is exactly the same as the compound matrix multiplication we used to calculate R in equation (II-2-9).

Since layer matrix a can be factored into $E \Lambda E^{-1}$, Abo-Zena further removed the phase terms in Λ out of a in order to control the exponentially growing terms when the phase velocity is less than a wave velocity (Abo-zena, equation 44). A similar technique can also be found in Kennett's (1974) 'phase-related' operation (ref. section 4.3). In such a reformulation, the square exponential terms are identically zero, as we have seen in deriving the compound matrix a. It is this procedure which makes the matrix method useful for very high frequencies.

Mantle Wave Simulation

A scheme by which the teleseismic body wave pulses from a seismic source are calculated, allowing for the effects of transmission through the mantle and crust, was given by Carpenter (1966) and Hudson (1969b). The calculation is divided into three parts: source crust response, mantle effect, and receiver crust response. The rays emitted from the source crust are allowed to travel through the homogeneous upper mantle, then enter the receiver crust and be recorded. Hudson (1969b) gave a detailed derivation providing both formalisms and approximating solutions. Here we will extend his result by using the compound matrix. Although both Carpenter (1966) and Hudson (1969b) start from Haskell (1964), our derivation is relatively simple and direct.

In the source crust, the waves from the source are reflected and refracted and finally enter the bottom halfspace as mantle waves following the equation:

$$\mathbb{K}_N = \mathbb{X}\mathbb{S} + \mathbb{R}\mathbb{B}_1 .$$

After expanding, setting $A_N'' = B_N'' = 0$ and $T_{r_1} = T_{z_1} = 0$, and substituting equation (II-2-7), we have

$$A_N' = [\, R\,|^{12}_{12}\, X_{3i}S_i + R\,|^{23}_{12}\, X_{1i}S_i - R\,|^{13}_{12}\, X_{2i}S_i \,] \,/\, R\,|^{12}_{12}$$

for P waves. This equation is equivalent to equation (3.3) of Hudson (1969b). Another form can easily be found by using

$$R^{-1}|^{ij}_{kl} = q_N(-1)^{i+j+k+l} R\,|^{\overline{kl}}_{\overline{ij}} .$$

The result is

$$A_N' = S_i\, R^{-1}|^{34}_{j4}\, X_{ji} \,/\, R^{-1}|^{34}_{34} ,$$

or, by $R^{-1} = Z^{-1} X^{-1}$

$$A_N' = S_i\, Z^{-1}|^{34}_{ij}\, X^{-1}_{j4} \,/\, R^{-1}|^{34}_{34} .$$

Similarly for SV waves, we have

$$B_N' = -S_i\, R^{-1}|^{34}_{j3}\, X_{ji} \,/\, R^{-1}|^{34}_{34}$$
$$= -S_i\, Z^{-1}|^{34}_{ij}\, X^{-1}_{j3} \,/\, R^{-1}|^{34}_{34} .$$

Inside the mantle, the waves as expressed in displacement form are $B_{N+1} = E_N \Lambda_N K_N$

or,

$$\begin{bmatrix} U_r \\ U_z \end{bmatrix}_{N+1} = \begin{bmatrix} -\frac{k}{\rho} e^{-\nu_\alpha z} A_N' + \frac{\nu_\beta k}{\rho} e^{-\nu_\beta z} B_N' \\ -\frac{\nu_\alpha}{\rho} e^{-\nu_\alpha z} A_N' + \frac{k^2}{\rho} e^{-\nu_\beta z} B_N' \end{bmatrix} = \begin{bmatrix} U_r^P + U_r^{SV} \\ U_z^P + U_z^{SV} \end{bmatrix}_{N+1}, \tag{II-3-5}$$

where z is measured from the deepest interface. The solutions are divided into P and S components corresponding to A_N' and B_N' respectively. The final solutions similar to equation (II-2-15) can easily be found following the procedures in section 2.2.

When waves pass through the mantle, the effect of the mantle can be simply included by adding geometric spreading factors for P and S motions respectively (Ben-Menahem and Singh, 1972). The wave fields expressed in equation (II-3-5) are decomposed into the compressional and shear components by using the angle ϑ which is the angle made by the downward vertical with the ray direction:

$$U^P = U_r^P \sin \vartheta_P + U_z^P \cos \vartheta_P$$
$$U^{SV} = U_r^{SV} \cos \vartheta_S - U_z^{SV} \sin \vartheta_S$$

where

$$\vartheta_P = \tan^{-1} \frac{k}{\nu_{\alpha_N}}$$
$$\vartheta_S = \tan^{-1} \frac{k}{\nu_{\beta_N}} .$$

U^P and U^{SV} are then multiplied by the spreading factors of P and SV waves, respectively, to take into account the mantle effect. At the bottom of the receiver crust, the P and SV waves can be resolved back into the r and z directions as follows:

$$U_r' = U^{P'} \sin \vartheta_P - U^{SV'} \cos \vartheta_S$$
$$U_z' = -U^{P'} \cos \vartheta_P - U^{SV'} \sin \vartheta_S ,$$

where the prime indicates the mantle waves which reach the receiver crust.

The effect of the receiver crust is easily taken into account by using the Haskell matrix R from the whole layer stacking. For our system, we just need to change $(J_{32} - J_{42})$ in Hudson's equation (7.7) to R_{22}, $(J_{31} - J_{41})$ to R_{21}, and evaluate the period equation by compound forms.

In the above discussion, we have shown how easy it is to extend our system. Many theories with difficult derivations come out simply by using the relation of Haskell matrices developed in section 2.1. More examples will be given in the later chapters.

2.4 Numerical Integration

Contour Integration

All of the integrals in equation (II-2-15) have the general form

$$\int_0^\infty f(k,\omega)\, J_n(kr)\, dk \; . \tag{II-4-1}$$

The evaluation of this integral is complicated because the function $f(k,\omega)$ has poles and branch points (Figure 2). The poles come from the zeros of the period equation, and the branch points from the radicals $\nu_{\alpha N}$ and $\nu_{\beta N}$ given in equation (II-1-5). To avoid dual values of these radicals, we introduce the branch cuts at the loci along which the real parts of $\nu_{\alpha N}$ and $\nu_{\beta N}$ are equal to zero. For the limiting case of real frequency and real layer velocities the two branch cuts, corresponding to $\nu_{\alpha N}$ and $\nu_{\beta N}$ respectively, collapse to form one branch cut along the real k-axis from the S branch point to the origin, then along the imaginary k-axis to $-i\infty$. The number of poles is finite and all of them are located on the real k-axis. The positions of poles determine the dispersion values on the ω-k domain.

The integration in equation (II-4-1) can be evaluated by applying Cauchy's theorem. This is a standard technique and is used, for example, by Ewing et al (1957), Harkrider (1964), Hudson (1969b), and Herrmann (1979) in their work. In order to deform the contour properly we need to apply a transformation which expresses the Bessel function in terms of the Hankel functions of the first and second kind:

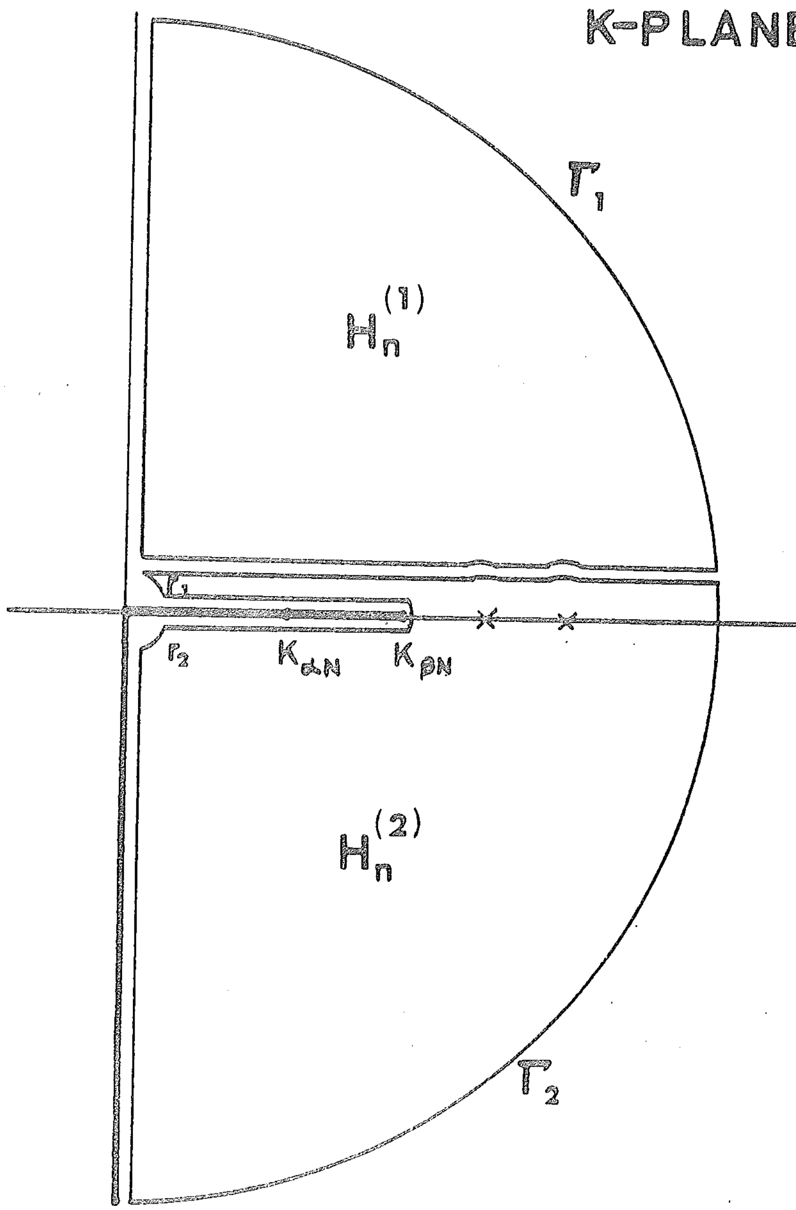


Figure 2. Contours in the complex k plane for evaluating the wavenumber integrals. The positions of the k_{α_N} and k_{β_N} branch points and the surface-wave poles ('X's') are indicated. Branch cuts are shown by thicker lines. Two circular arcs, γ_1 and γ_2 , sourround the possible Hankel function pole at $k = 0$.

$$J_n(kr) = \frac{1}{2}\left[H_n^{(1)}(kr) + H_n^{(2)}(kr) \right] .$$

The integral path can be deformed into the first and fourth quadrants, since $H_n^{(1)}$ and $H_n^{(2)}$ are analytic in the first and fourth quadrants, respectively, of the complex k plane. These two contours can further be combined into one as shown in Figure 2. Herrmann (1979) finally expressed the contour integration in the form:

$$\begin{aligned} \int_0^\infty f(k,\omega)\, J_n(kr)\, dk &= -\pi i \sum Res\, f(k,\omega)\, H_n^{(2)}(kr) \\ &+ i \int_0^{k_{\beta_N}} I_m\, f_+(k,\omega)\, H_n^{(2)}(kr)\, dk \qquad \text{(II-4-2)} \\ &+ \left(\frac{1}{\pi}\right) \int_0^\infty \left[f_+(i\tau,\omega)\, exp(-in\pi/2) \right. \\ &\left. + f_-(-i\tau,\omega)\, exp(in\pi/2) \right] K_n(\tau r)\, d\tau , \end{aligned}$$

where the + or - subscripts indicate that ν_{α_N} , ν_{β_N} > 0 or < 0, respectively, be used to evaluate the function. K_n is the modified Bessel function which decreases exponentially with increasing argument.

At each different frequency, when wavenumber k approaches zero, there might be some singularity introduced by the Hankel function. Wang and Herrmann (1980) realized that the solution (II-4-2) is correct only if $f(k,\omega) = O(k^{n+1})$ as k approaches zero where n is the order of the Bessel function. Otherwise the singularity of the Hankel function at k = 0 contributes to the integral. After carefully examining the k terms in the response function, we find that such a Hankel singularity only occurs in the components of the near-field displacement of a double-couple source, for which $f(k,\omega) = O(k^0)$ for n = 1 and $f(k,\omega) = O(k^1)$ for n = 2.

Figure 2 shows the proper contour to be used for contour integration, where we have made a small circle to avoid the singularity at k = Ø. One cannot contract the small contour about the origin since the Hankel function singularity is still present at the lower limits of the branch line integrals. An interesting way to overcome this is to use a known integral whose behavior at k = Ø is the same as that of the functions in equation (II-4-1). Using an integral from Harkrider (1976) :

$$\int_0^\infty \frac{k_\upsilon\, J_1(kr)\, dk}{\nu_\upsilon} = [\, 1 - exp(-ik_\upsilon r)\,] / ir \;,$$

where $\nu_\upsilon^2(k,\omega) = k^2 - k_\upsilon^2$. The integral for n = 1 becomes

$$\int_0^\infty f(k,\omega)\, J_1(kr)\, dk = -\pi i \sum Res\, [\, f(k,\omega)\, H_1^{(2)}(kr)\,]$$

$$+ i \int_0^{k_{\beta_N}} I_m\, [f_+(k,\omega) + \frac{k_\upsilon}{\nu_\upsilon} I_m f_+(0,\omega)]\, H_1^{(2)}(kr)\, dk$$

$$+ \frac{2}{\pi} \int_0^\infty I_m\, [f_+(i\tau,\omega) + \frac{k_\upsilon}{\nu_{\upsilon_+}(i\tau)} I_m f_+(0,\omega)]\, K_1(\tau r)\, d\tau$$

$$+ \frac{1}{r} \text{Re } f_+(0,\omega) \tag{II-4-3}$$

$$+ i\frac{I_m f_+(0,\omega)}{r} [1 - exp(ik_v r)] \ .$$

To evaluate the near field integrals for n = 2, a careful examination of the integrands revealed that the integrand could easily be factored into the form $f(k,\omega) = kg(k,\omega)$. This contour integration used the other integral of Harkrider (1976):

$$\int_0^\infty \frac{k_v k \ J_2(kr)}{\nu_v} dk = -\frac{k_v}{r} exp(-ik_v r) + 2[1 - exp(ik_v r)] / ir^2 \ .$$

Using this integral and performing the contour integration;

$$\int_0^\infty f(k,\omega) J_2(kr) \, dk = -\pi i \sum Res \, [k \ g(k,\omega) H_2^{(2)}(kr)]$$

$$+ i \int_0^{k_{\beta_N}} I_m \, [g_+(k,\omega) + \frac{k_v}{\nu_v} I_m g_+(0,\omega)] \ k \ H_2^{(2)}(kr) \, dk$$

$$+ \frac{2}{\pi} \int_0^\infty I_m \, [g_+(i\tau,\omega) + \frac{k_v}{\nu_v(i\tau)} I_m g_+(0,\omega)] \ \tau K_2(\tau r) \, d\tau$$

$$+ \frac{2}{r^2} \text{Re } g_+(0,\omega) \tag{II-4-4}$$

$$+ \frac{I_m g_+(0,\omega)}{r} [k_v \exp(-ik_v r) + i\frac{2}{r}(1 - exp(-ik_v r))] \ .$$

In both equations (II-4-3) and (II-4-4), the lower limits of integration equal zero because the functions $f(k,\omega)$ for n = 1 and $g(k,\omega)$ for n = 2 are even functions in k. Therefore the integrands in the branch line integrals are identically zero as the lower limit of integration goes to zero. The choice of the parameter k_v is somewhat arbitrary. For the SH terms, $k_v = k_{\beta_N}$ is used while $k_v = k_\alpha$ from equation (II-4-5) is used for the P-SV terms. This point will be further discussed later.

Numerical Integration

The solutions as shown in the integral form of equation (II-4-2) consist of two contributions: the pole residue contribution and the branch line integrals. Generally speaking, the pole contribution gives rise to surface waves and the branch line integral yields most of the body waves. The pole and its residue will be discussed using normal mode theory in the next chapter. Later in chapter IV, the branch line contribution will be examined using a leaky mode approach. Here we just give a brief discussion of these two contributions, and attempt to improve the numerical integration technique.

To evaluate the integral

$$\int_0^{k_{\beta_N}} \{ I_m f(k,\omega) \} H_n^{(2)}(kr)\, k^m \, dk \ ,$$

knowledge of the behavior of the function $I_m f(k,\omega)$ is essential. Figure 3 shows the variation of these functions along the real branch axis at a frequency of 1 Hz for a point source at a depth of 10 km in the central

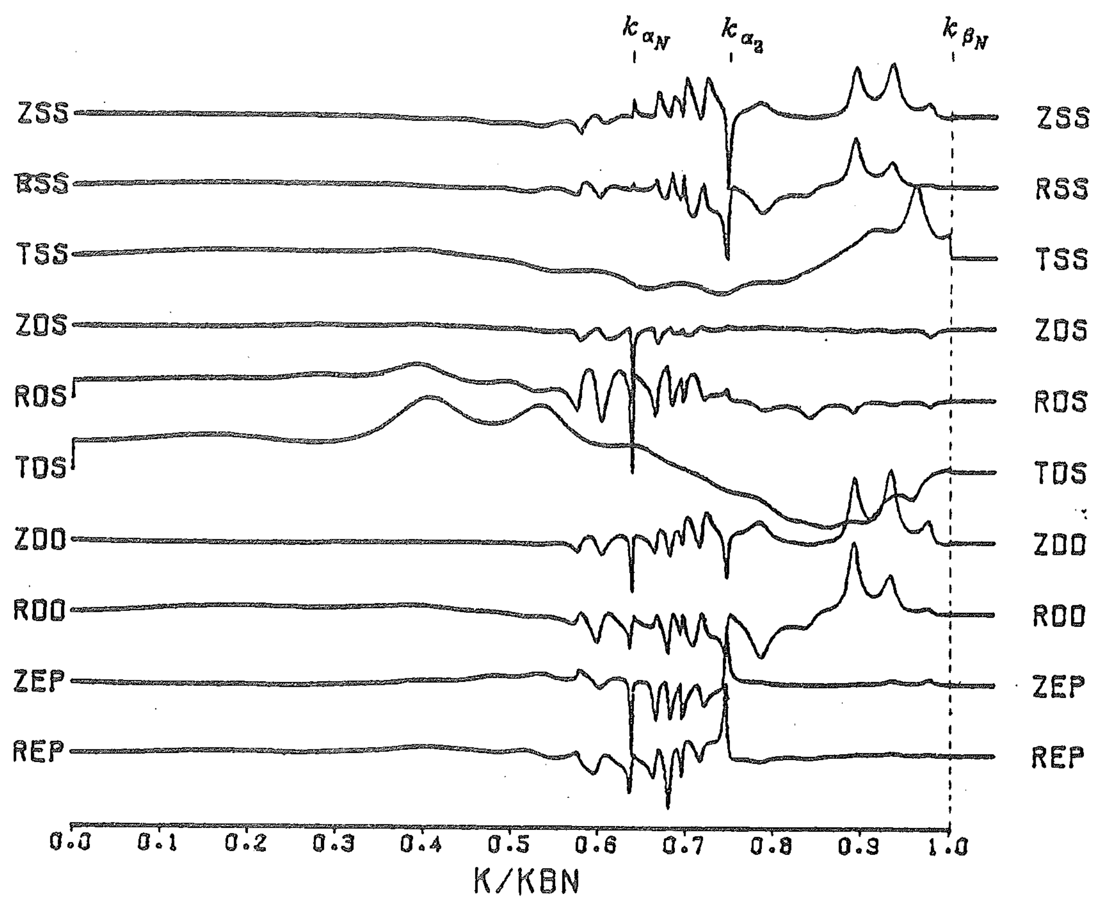


Figure 3. The responses of integrands in equation (II-2-15) along the real k-axis branch cut. The central United States (CUS) earth model, a source depth of 10 km and a frequency of 1.0 Hz, are used.

United States (CUS) model (Nuttli et al, 1969; Herrmann, 1978a) of Table 1. The symbols indicate that the functions plotted arise from the corresponding integrands of far-field terms in equation (II-2-15). It can be noticed that the peaks and troughs of the excursions, corresponding to P-SV and SH respectively, are aligned, although the magnitude might vary. This is caused by some factors independent of the source type. Another feature seen in Figure 3 is that the tangential components, TSS and TDS, are substantially smoother than any of the P-SV function. The P-SV terms (Z and R components) oscillate rapidly near both $k_{\alpha N}$ and $k_{\beta N}$ branch points, indicating that they should be sampled with care. Note also the behavior of the RDS and TDS plots as k goes to zero. These non-zero limits are the reason for the involved contour integration required in equations (II-4-3) and (II-4-4).

Figure 4 shows the variation of integrands for the ZSS component with frequencies between 0.10 and 10.0 Hz. The point source is at a depth of 10 km in the simple crust model (SCM) of Table 1. The requirements for adequate sampling of the integrand for proper numerical integration at high frequencies are obvious. Given the frequency f, we found it adequate to sample the region between $(0, k_{\beta_N})$ by $50+200*f$ points. This sampling represents one of the most time consuming aspects of our computations. Such calculations set practical

TABLE 1

Earth Models

Thickness (km)	P vel (km/sec)	S vel (km/sec)	Density (g/cm^3)
	Simple Crustal Model		
40	6.15	3.55	2.8
–	8.09	4.67	3.3
	Central U. S. Model		
1	5.00	2.89	2.5
9	6.10	3.52	2.7
10	6.40	3.70	2.9
20	6.70	3.87	3.0
–	8.15	4.70	3.4

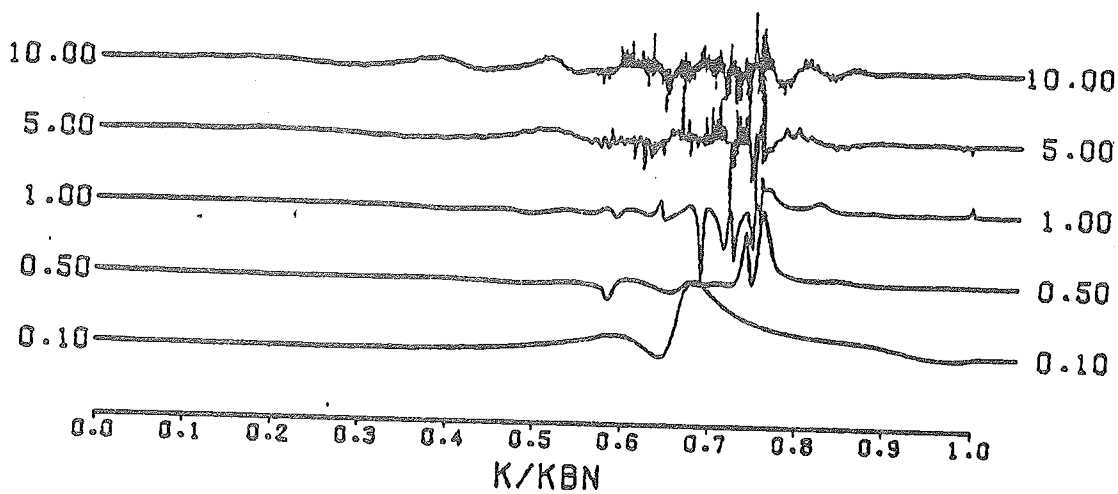


Figure 4. The real k-axis wavenumber responses of ZSS component along the real branch cut as a function of frequency between 0.1 and 10.0 Hz. The simple crust model (SCM) and a 10 km deep source are used.

limits to this form of numerical integration.

To alleviate any numerical problems near the P and S branch points, a change of variable of integration is introduced following Fuchs and Müller (1971):

$$k = \begin{cases} k_\alpha \sin\gamma & \gamma = [0, \frac{\pi}{2}] \quad \text{for } 0 < k \leq k_\alpha \\ \frac{k_\alpha + k_{\beta_N}}{2} + \frac{k_\alpha - k_{\beta_N}}{2} \cos\gamma & \gamma = [0, \pi] \quad \text{for } k_\alpha < k \leq k_{\beta_N} \end{cases} \qquad \text{(II-4-5)}$$

where k_α is chosen as an average of $k_{\alpha_n} * d_n$ over all upper layers. This formulation permits the solution of the halfspace problem as a subset of the layered halfspace problem by removing the possible singularities due to the branch points at $k_{\alpha N}$ and $k_{\beta N}$. For a layered medium, the transformation weights out the possibility of a surface wave pole at the $k_{\beta N}$ branch point interacting with the real branch line integral. The success of this particular transformation is evident from the quality of the synthetic seismograms presented later. For SH integrals, the transformation $k = k_{\beta_N} \sin\gamma$ for $\gamma = (0, \pi/2)$ is used.

The residues due to the poles of the integrand occur at the zeroes of the period equation. The number of poles, or surface wave modes, increases with frequency. For example, the SCM model of Table 1 has one Rayleigh wave pole at the frequency of 0.01 Hz, and twenty poles at the frequency of 2 Hz. The number of poles increases almost linearly with frequency. For the dispersion studies, the positions of poles need only be grossly located compared to the precision requirements for seismogram synthesis.

Figures 5 and 6 display the dispersion relations in the frequency-wavenumber plane and phase velocity-period plane, respectively, for a complex oceanic model. A stair-like dispersion pattern is obvious, which usually causes difficulty for pole searching. To overcome this, a pole searching technique was developed. Since the dispersion curves vary more gently in the frequency-wavenumber domain than in the phase velocity-period domain, the pole location will be traced in the ω-k plane.

We start with a very fine search between $k = k_{\beta_N}$ and $k = k_{\beta_{min}}$ (for Rayleigh wave use $k = k_{\beta_{min}}/0.88$ to include the fundamental mode) at the two highest frequencies of interest. At this stage a dense search technique with an interval halving refinement is used to find zero crossings. The poles at the next lower frequency could, of course, be found by repeating the same procedures. However, we can save computation time by using the fact that for most earth models, the phase velocity of a given mode always increases with decreasing frequency. For a given mode we known that

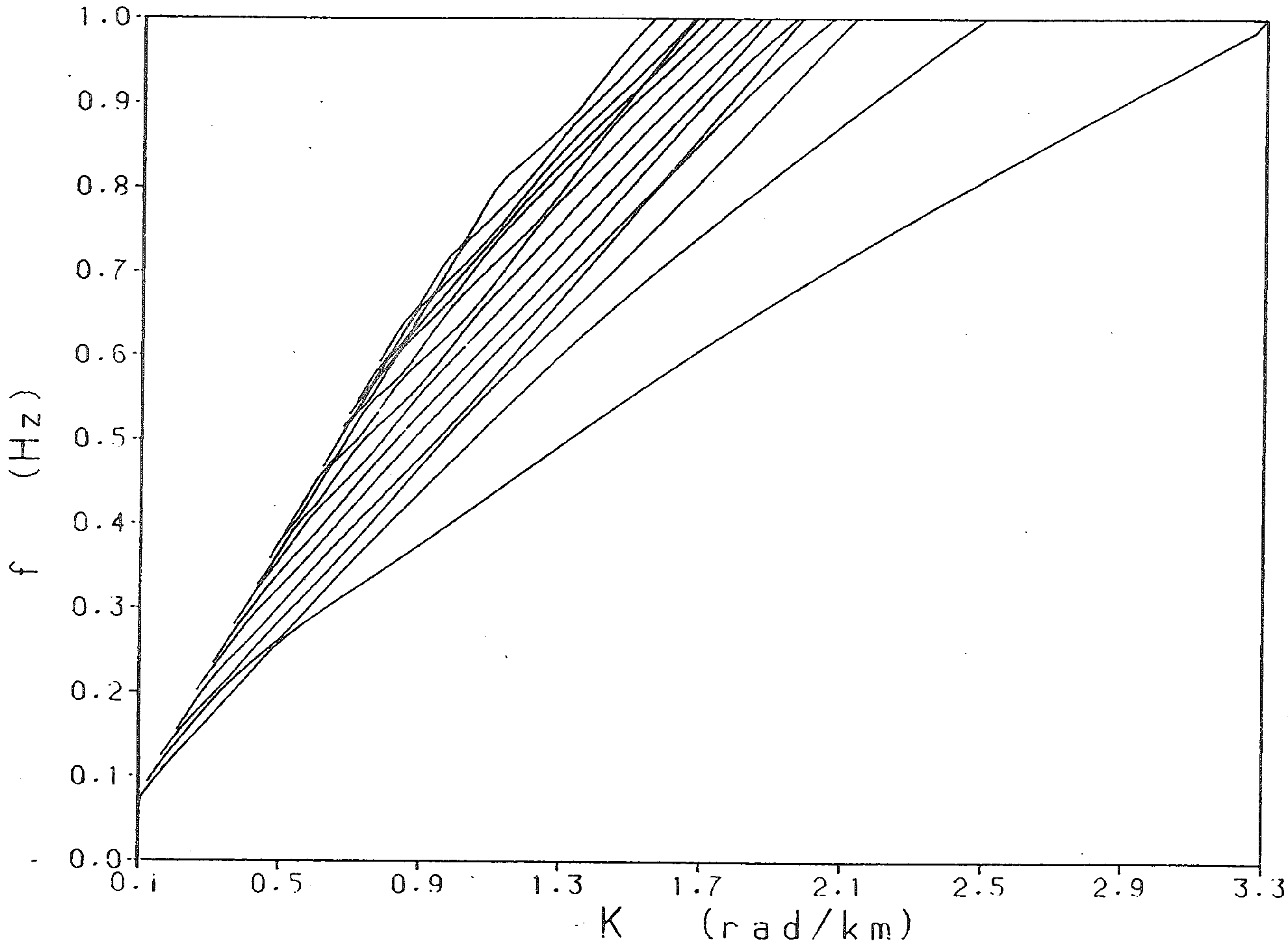


Figure 5. The dispersion curves for Rayleigh waves as expressed in the frequency and wavenumber plane for a 30 layer oceanic model.

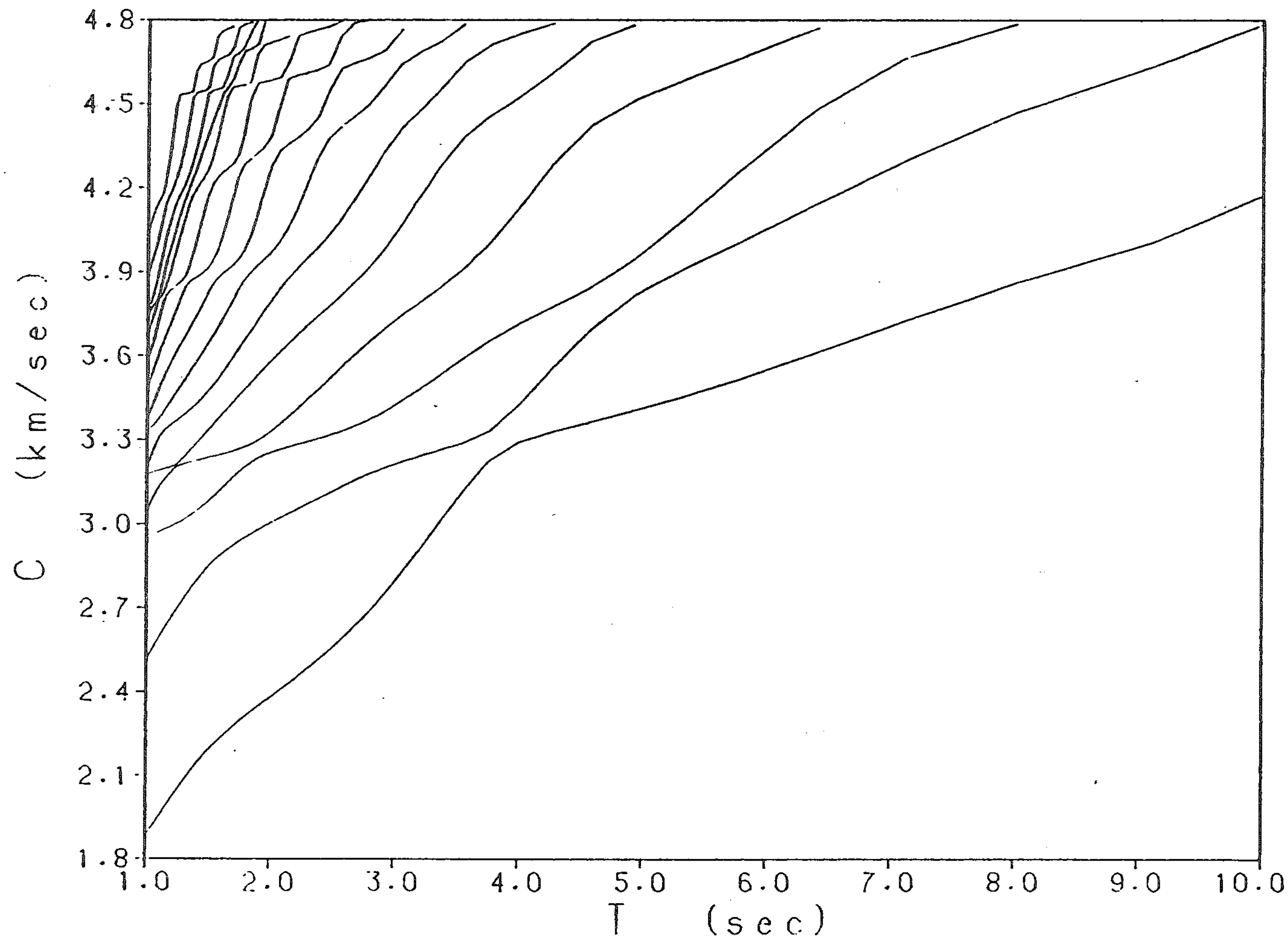


Figure 6. The dispersion curves for Rayleigh waves as expressed in the phase velocity and period plane. The same earth model as Figure 5 is used.

$$\omega = k\,c\ ,$$

and that

$$\Delta k = \frac{\Delta\omega}{c} - k\frac{\Delta c}{c}\ . \qquad \text{(II-4-6)}$$

This can be used to estimate the pole position of a mode from its position at slightly higher frequencies. The previously calculated pole positions at two higher frequencies are used to give Δc. The search at the present frequency begins with the lowest order mode at $k_{new} = k_{old} + \frac{\Delta\omega}{c}$. This guarantees that a particular mode is not overshot.

Next, several fractions of the increment $k\frac{\Delta c}{c}$ are added to k_{new} to find the place where the sign of the period equation changes, at which time the interval-halving method is used to refine the zero. The computational efficiency as well as the results are found to be substantially improved, since the modes are now followed rather than searched at each frequency.

The numerical evaluation of the imaginary axis branch line integrals is the same as that given by Herrmann (1979). The Gauss-Laguerre integration rule was used since P-SV, SH functions vary harmonically and since the $K_n(\tau r)$ decayed exponentially;

$$\int_0^\infty f(i\tau,\omega)\,K_n(\tau r)\,d\tau = \sum_{k=1}^{m} w_k[f(i\frac{x_k}{\tau})]\,[x_k e^{x_k} K_n(x_k)]\,\frac{1}{x_k r}$$

$$\int_0^\infty f(i\tau,\omega)\,K_n(\tau r)\,\tau\,d\tau = \sum_{k=1}^{m} w_k[f(i\frac{x_k}{r})]\,[x_k e^{x_k} K_n(x_k)]\,\frac{1}{r^2}\,, \qquad \text{(II-4-7)}$$

where w_k is the weight and $x = \tau r$. Because of the r term in equation (II-4-7), the imaginary branch line integral only becomes important at short distances. For computation, the integrand function equation (II-2-7) was used rather than compound matrix formulation (II-2-8) because there are no exponential terms involved. An $m = 100$ order Gauss-Laguerre integration rule is used, which should be valid at radial distances as close as one-half source depth.

Synthetic Seismograms

After the values of $u_z(r,\vartheta,0,\omega)$, $u_r(r,\vartheta,0,\omega)$, and $u_\vartheta(r,\vartheta,0,\omega)$ are calculated at several discrete frequencies, we take the inverse Fourier transform with respect to frequency to form the time histories. The source time function with spectrum $s(\omega)$ should be taken into account at the same time:

$$u(r,\vartheta,0,t) = \int_{-\infty}^{\infty} s(\omega)\,u(r,\vartheta,0,\omega)\,exp(i\omega t)\,d\omega/2\pi\,.$$

Since the fast Fourier transform (FFT) is used to approximate the Fourier integral, the source time function given by Herrmann (1979)

$$2\tau s(t) = \begin{cases} 0 & t \le 0 \\ \frac{1}{2}(\frac{t}{\tau})^2 & 0 < t \le \tau \\ -\frac{1}{2}(\frac{t}{\tau})^2 + 2(\frac{t}{\tau}) - 1 & \tau < t \le 3\tau \\ \frac{1}{2}(\frac{t}{\tau})^2 - 4(\frac{t}{\tau}) + 8 & 3\tau < t \le 4\tau \\ 0 & 4\tau < t \end{cases} \qquad \text{(II-4-8)}$$

is used. In the context of dislocation theory, $s(t)$ is proportional to the velocity of the dislocation, or equivalently the far-field displacement time history in an infinite medium. The time histories generated by such an impulse will be those of ground velocity. The Fourier amplitude spectrum of this pulse is enveloped by f^0 and f^{-3} asymptotes which intersect at a corner frequency of $f_c = 1/(4.36*\tau)$ (Herrmann and Wang, 1979). To avoid numerical noise problems with the FFT, τ is taken to be an integral of sampling interval, Δt .

The above numerical technique was tested by generating theoretical seismograms using the models listed in Table 1. Figures 7, 8, and 9 illustrate some high quality seismograms from a vertical dip-slip source in the SCM model for three different components, respectively. A source depth of 10 km, seismic moment of 1.0 E +20 dyne-cm, source-time function with τ = 0.4 sec, and a frequency range from 0.0 to 1.25 Hz are used. The seismograms at distances less than 100 km include both the near- and far-field terms; beyond this dis-

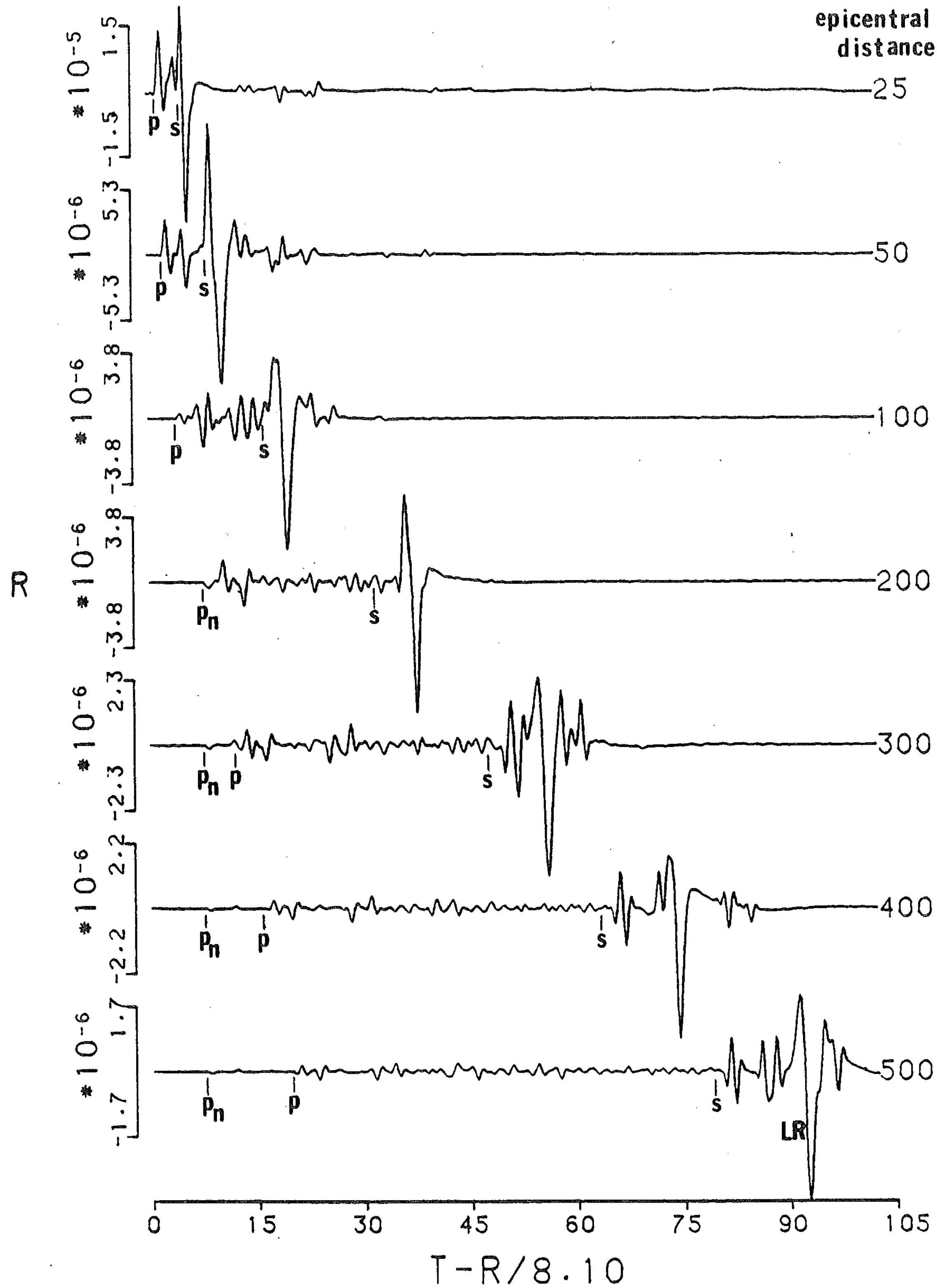


Figure 7. Radial component velocity time histories (RDS) due to a vertical dip-slip dislocation source at a depth of 10 km in SCM model. A source time function with τ=0.5 sec and seismic moment of 1.0E20 dyne-cm are used.

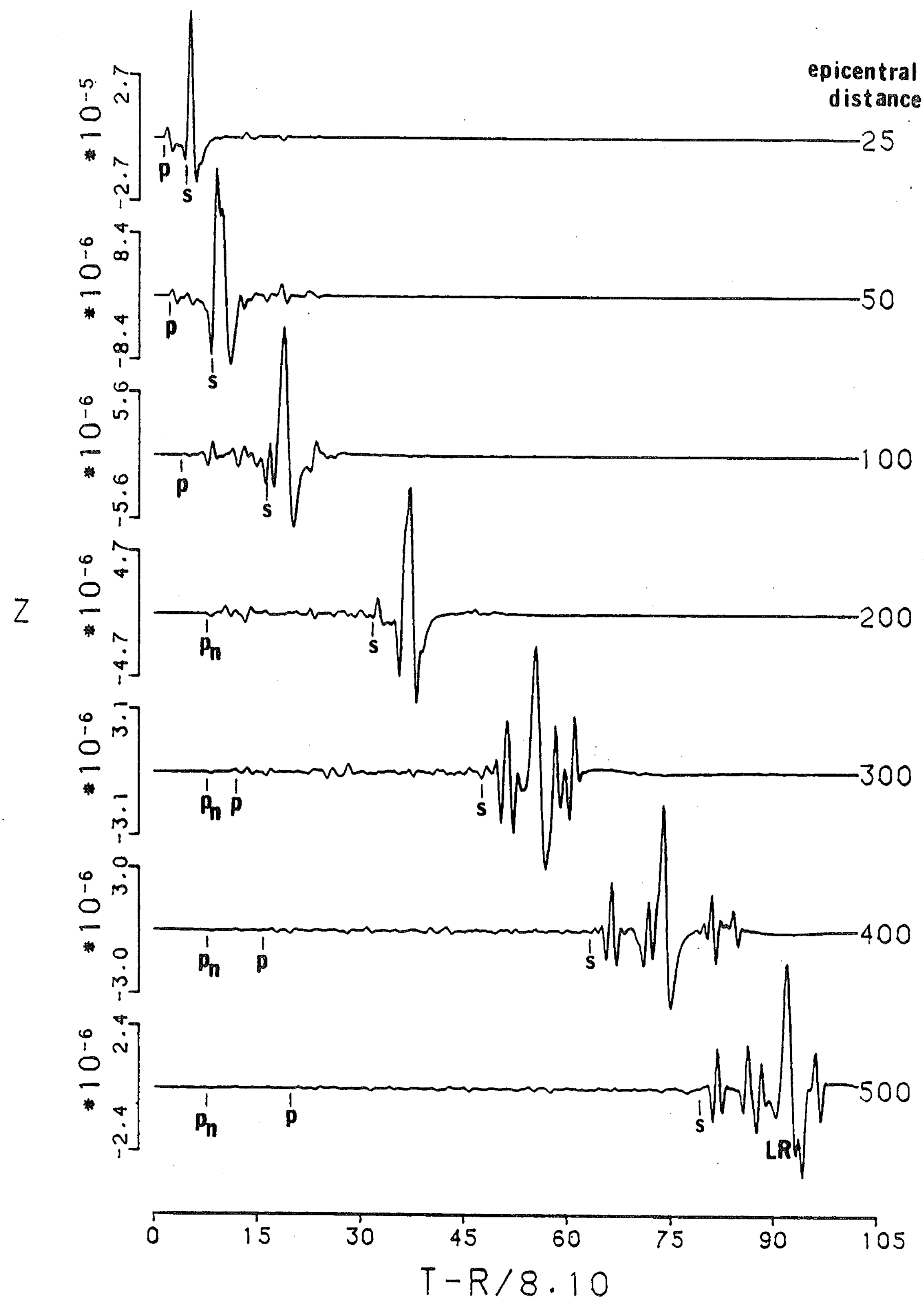


Figure 8. Results corresponding to Figure 7 but for the vertical component.

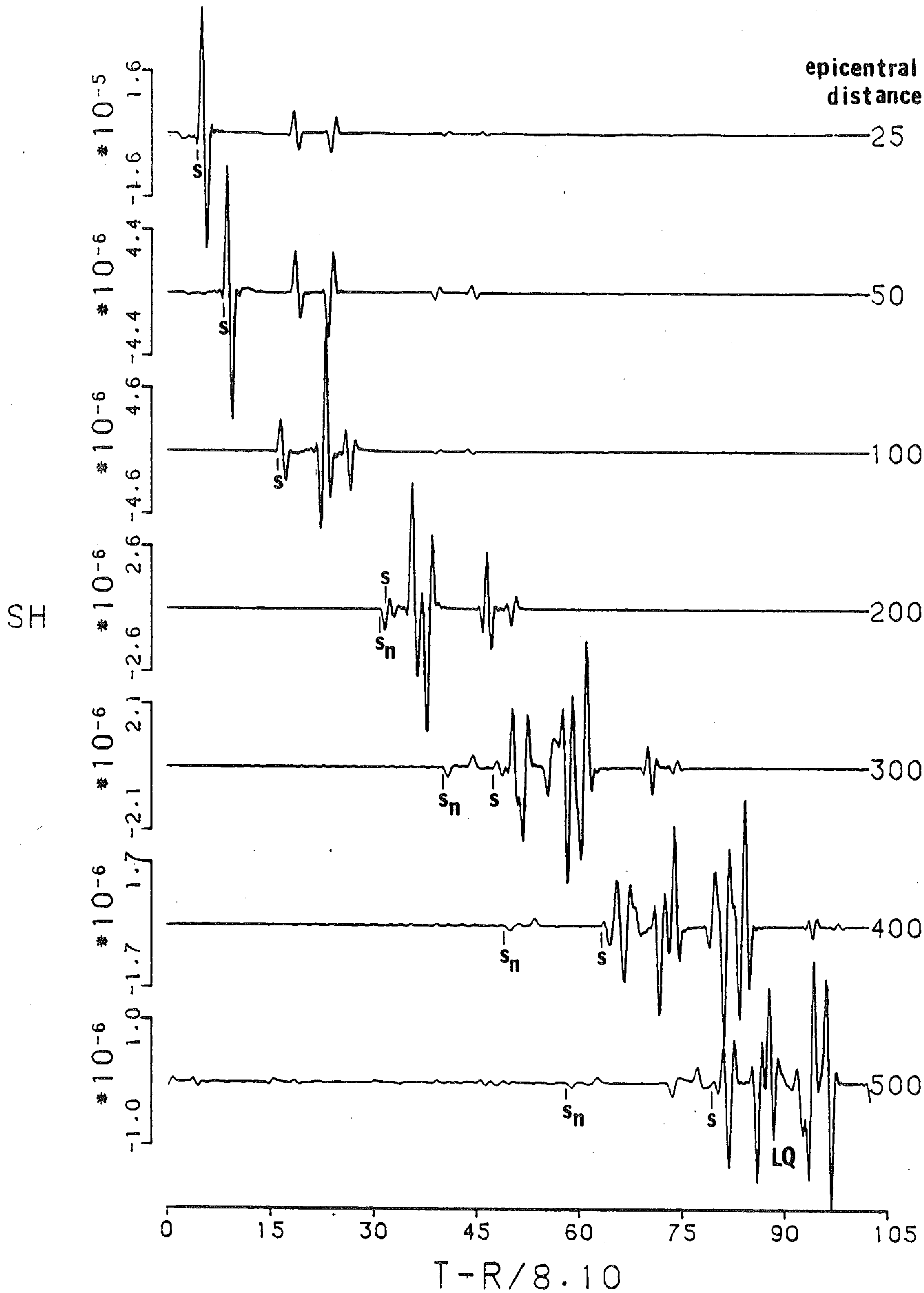


Figure 9. Results corresponding to Figure 7 but for the tangential component.

tance only the far-field terms are used. It is noted immediately that the body and surface phases are very clear. At large distances, there are multiple reflections following Pn, the first arrival, which last until the onset of the surface waves. Rayleigh-wave groups are well developed, even at short distances, in distinction to the Love-wave group which becomes apparent only at ranges larger than the S-wave critical distance (about 81.9 km). At the distance of 500 km, an interpolation of the pole contribution with one-half the frequency spacing was applied to double the time window (Kennett, 1980). At this distance, only the first half of the time series is plotted.

Figure 10 displays the radial component seimograms due to a 45° dip-slip source of 1 km depth in the CUS model of Table 1. The reverberation within the top layer provides a wave guide which generates large surface waves. The well-developed dispersed groups consist mostly of fundamental mode waves, which might have been greatly attenuated in the real earth structure. Some higher mode Lg wave arrivals prior to the dispersed wave train can easily be seen.

Figure 11 shows the effect of focal depth. The model used is the CUS model of Table 1, and the source type is vertical strike-slip. The epicentral distance is kept at 100 km. Numbers at the end of each seismo-

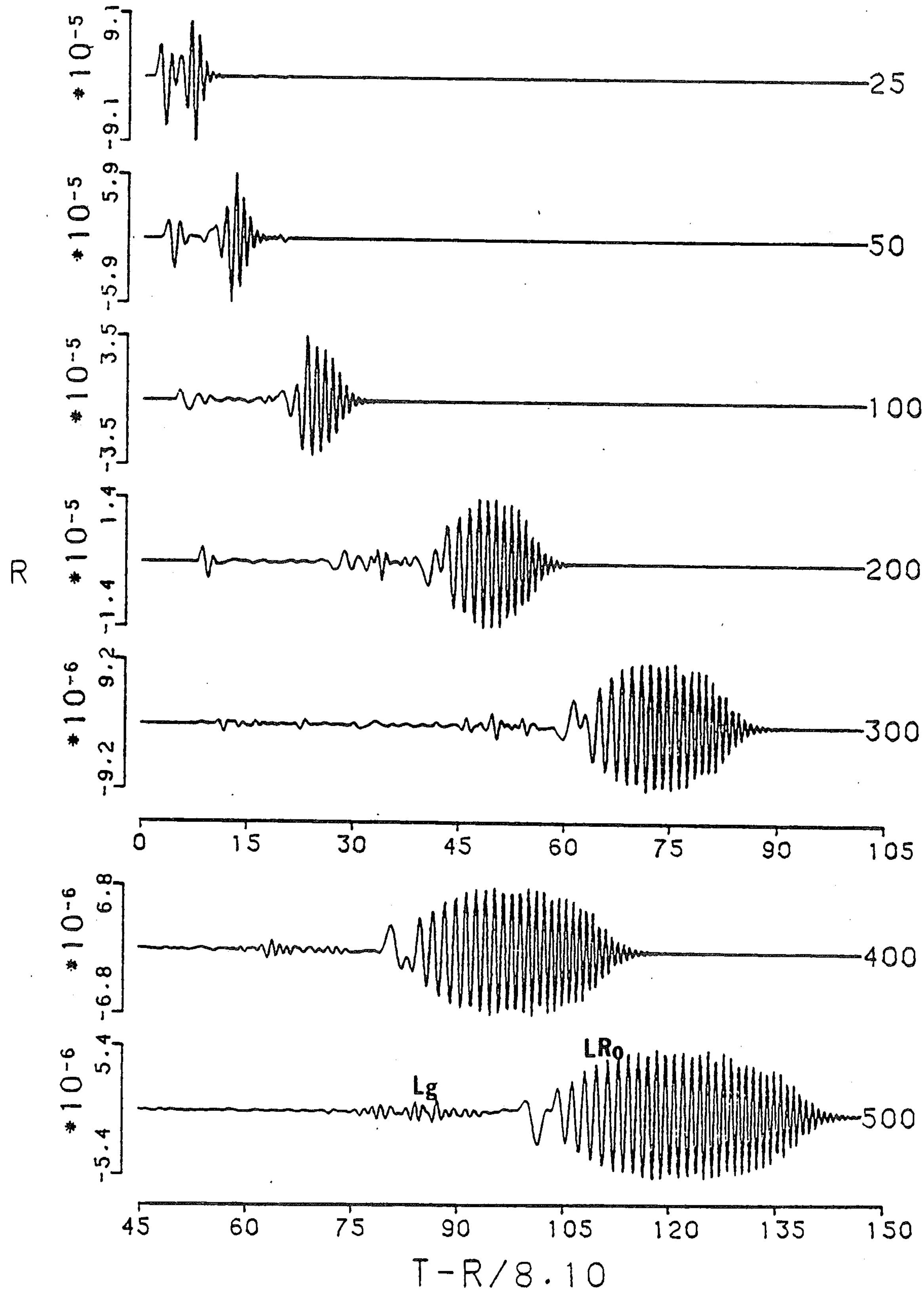


Figure 10. Radial component velocity time histories (RDD) due to a 45° dip-slip dislocation source at a depth of 10 km in CUS model. Other parameters are the same as in Figure 7.

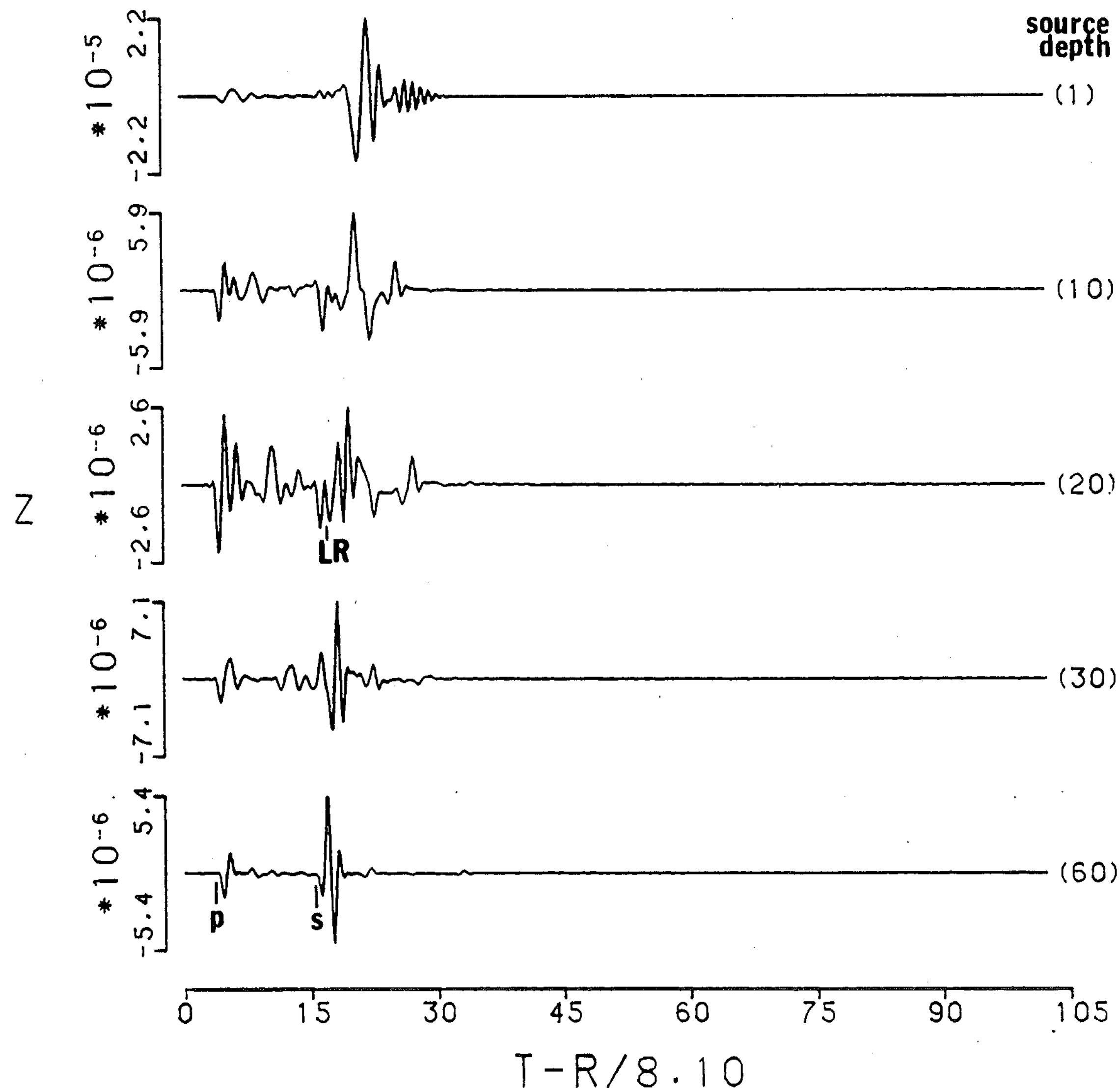


Figure 11. Vertical component velocity time histories (ZSS) due to a vertical strike-slip source in CUS model. The sources are buried at different depth as indicated at the end of each seismogram. Epicentral distance is kept at 100 km. Other parameters are the same as in Figure 7.

gram are the source depths. It seems that for shallow earthquakes the surface wave is dominant. When the source is deeper the seismogram becomes complex due to the increase of the body waves with respect to the surface waves. When the source is very deep the seismogram becomes simple again, because it consists mostly of body waves. Surface waves have not yet had time to develop. Note that the source at depth 60 km is already in the halfspace.

CHAPTER III

SURFACE WAVE - NORMAL MODE STUDY

The application of eigenfunction theory to the study of surface waves is an important step which incorporates modern mathematics into the investigation of wave propagation in the earth. Using Green's theory as a tool, the response functions in the transformed domain can be defined along mutually perpendicular directions of chosen coordinates. By the imposition of boundary conditions, the values of response functions are discretized and yield specific modes of the surface wave. Because of this, the theory is also called normal mode theory. Orthogonality and normalization, two most useful procedures in eigenfunction theory, further produce the final solutions in a form which is easy to use. The reason for such a discrete excitation arises because surface waves are a type of trapped waves which reflect back and forth in the layers, and decay exponentially in the halfspace. A consequence of this system is the conservation of its total energy. Takeuchi and Saito (1972) used the calculus of variations to derive quantities concerning the derivatives of the conservative energy. These partial derivatives are somewhat sensitive to the properties of the model and can be used to invert for an earth model from surface wave dispersion data.

The numerical application of eigenfunction theory was discussed by Bolt and Dorman (1961) and Takeuchi and Saito (1972). They chose two independent solutions along two perpendicular directions as initial values and used the Runge-Kutta method to integrate over layers. The boundary condition at the free surface is imposed in order to regulate the variation to final solutions. Because of exponential growth of several terms, this method fails to fit the special case of very high frequencies. In this chapter, we will derive a new method to calculate the eigenfunctions. The main effort is directed toward relating the eigenfunction theory to the general layer matrix method. As a result of the symmetry properties of Haskell matrices revealed in the last chapter, the eigenfunctions as well as the energy integrals can be expressed in closed, analytic, forms. Some well-known formulae are revised and rigorously proved using the new framework. The overall objective of this chapter is the calculation of high frequency signals in any plane layered model.

3.1 Haskell Matrices for the Layers

The eigenfuction problem for the generation of surface waves in plane layered media is an important topic in the theory of seismogram synthesis. After Lamb's famous study, many authors have explored this problem using different approaches (Alterman et al, 1959; Keilis-Borok, 1960; Saito, 1967; Takeuchi and Saito, 1972; Harvey, 1981). Here we will start with the solutions for displacements derived in the last chapter (equation II-2-8), and arrive at a formulation using eigenfunction values at the source depth. The reason we need the eigenfunctions rather than the Haskell's matrices X or Z is because these eigenfunctions can be used to determine the energy which is conserved for surface waves traveling in the layers. Such an energy system is permitted to be perturbed to derive other properties. For example, quantities such as group velocity, dissipation functions, and amplitude factors, which arise from small perturbations of elastic parameters, can be obtained by the variation of total energy (Jeffreys, 1961; Harkrider and Anderson, 1966). In this section, a new formulation for evaluating the eigenfunction values at any depth, which was suggested by Dr. D. G. Harkrider (personal communication), will be presented. This formulation is not only computationally stable and precise, but is consistent with the system derived in the last chapter.

Now, let us discuss the eigenfunctions in the layers. The eigenfunction at any depth can be determined from the eigenfunctions at the free surface by the layer matrices in between

$$B_m = Z\, B_1 \qquad \text{(III-1-1)}$$

where

$$\mathbf{Z} = a_m a_{m-1} \cdots a_1 \ .$$

P-SV Eigenfunctions

As the first step, we consider the P-SV case. Expressing equation (III-1-1) in index form:

$$B_{m_k} = Z_{k1} U_{r_1} + Z_{k2} U_{z_1} \ , \qquad \text{(III-1-2)}$$

where we have used the surface boundary conditions which require the stresses to vanish at the free surface. For ease of expression, we will use the indices m, n, or N for the layers, and the indices i, j, k, or l for the components in the corresponding vector or matrix which have values 1 to 4 for P-SV waves and 5, 6 for SH waves. Normalizing by the surface z-component displacement U_{z_1} , equation (III-1-2) becomes

$$\overline{B}_{m_k} = B_{m_k} / U_{z_1} = Z_{k1} (U_{r_1}/U_{z_1}) + Z_{k2} \ . \qquad \text{(III-1-3)}$$

U_{r_1}/U_{z_1} is the ellipticity at the surface, which we already proved to be independent of the source and to possess several equivalent forms:

$$\begin{aligned} \varepsilon \equiv \frac{U_{r_1}}{U_{z_1}} &= - \frac{R_{12}}{R_{11}} = - \frac{R_{22}}{R_{21}} \\ &= - \frac{R\,|^{12}_{23}}{R\,|^{12}_{13}} = - \frac{R\,|^{12}_{24}}{R\,|^{12}_{14}} = \frac{R^{-1}|^{13}_{34}}{R^{-1}|^{23}_{34}} = \frac{R^{-1}|^{14}_{34}}{R^{-1}|^{24}_{34}} \ . \end{aligned} \qquad \text{(III-1-4)}$$

Using the last of equation (III-1-4), $R^{-1}|^{34}_{34} = 0$ (period equation), and $\mathbf{R} = \mathbf{X}\mathbf{Z}$, equation (III-1-3) becomes

$$\overline{B}_{m_k} = Z_{k1}\,(R^{-1}|^{14}_{34} \,/\, R^{-1}|^{24}_{34}) + Z_{k2}$$
$$= [Z_{k1}\,R^{-1}|^{14}_{34} + Z_{k2}\,R^{-1}|^{24}_{34} + Z_{k3}\,R^{-1}|^{34}_{34} + Z_{k4}\,R^{-1}|^{44}_{34}] \,/\, R^{-1}|^{24}_{34}$$
$$= [Z_{kp}\,R^{-1}|^{p4}_{34}] \,/\, R^{-1}|^{24}_{34}$$
$$= [Z_{kp}(R^{-1}_{p3}R^{-1}_{44} - R^{-1}_{p4}R^{-1}_{43})] \,/\, R^{-1}|^{24}_{34}$$
$$= [(Z_{kp}Z^{-1}_{pr})\,X^{-1}_{r3}\;Z^{-1}_{4l}\,X^{-1}_{l4} - (Z_{kp}Z^{-1}_{ps})\,X^{-1}_{s4}\;Z^{-1}_{4l}\,X^{-1}_{l3}] \,/\, R^{-1}|^{24}_{34}$$
$$= \frac{Z^{-1}_{4l}\,X^{-1}|^{kl}_{34}}{R^{-1}|^{24}_{34}} \quad . \qquad \text{(III-1-5)}$$

Equation (III-1-5) includes the inverses of Haskell's matrices, which can be related to normal matrices by using the symmetry properties of compound matrix described in Appendix C:

$$Z^{-1}_{jl} = (-1)^{j+l}\,Z_{5-l,5-j}$$
$$X^{-1}|^{kl}_{34} = q_N(-1)^{k+l+1}\,X|^{12}_{5-l,5-k}$$
$$R^{-1}|^{24}_{34} = q_N(-1)\,R\,|^{12}_{13} \quad .$$

Using these, equation (III-1-5) can be written in another form:

$$\overline{B}_{m_k} = \frac{(-1)^{k+1}\,X|^{12}_{5-l,5-k}\,Z_{5-l,1}}{-\,R\,|^{12}_{13}} \quad ,$$

by substituting 5-k=i 5-l=j,

$$\overline{B}_{m_{5-i}} = \frac{(-1)^i\,X|^{12}_{ij}\,Z_{j1}}{R\,|^{12}_{13}} \qquad \text{(III-1-6)}$$

which is equivalent to

$$X|^{12}_{ij}\,Z_{j1} = (-1)^i\,R\,|^{12}_{13}\,\overline{B}_{m_{5-i}} \quad . \qquad \text{(III-1-7)}$$

Equation (III-1-6) will be the form chosen for application.

From equation (III-1-6), we find that eigenfunctions at any depth can be calculated by a formula similar to equation (II-2-8), which has already been shown to be computationally stable and accurate. This point is worthwhile emphasizing. Since all of the Haskell matrices, their compound forms, and all of the boundary conditions are involved, this formula has attached to it many numerical and theoretical advantages. For example, the squared exponentially growing terms are suppressed by compound matrices. The models used are not restricted by the number of layers or layer thicknesses. The frequencies are permitted to go very high, say 200 Hz, as long as care is taken in evaluating the compound matrices. At the same time the efficiency and accuracy of calculation are all improved.

If the eigenvalues have already been found from another calculation, equation (III-1-6) can be used in the following way. First, we insert the source depths, which might be several different values, into the velocity structure as interfaces. To find the eigenfunctions at these interfaces, the calculation is started from the bottom of the layer stack upward. At every interface, $X|_{ij}^{12}$ is calculated and stored. When reaching the surface, X is transformed into R , and $R|_{13}^{12}$ is obtained. Next, a downward procedure is taken to find

Z_{j1}. After combining with the stored $X|_{ij}^{12}$ at that depth, we obtain the eigenfunction B by equation (III-1-6). It is obvious that in these procedures the layer matrix a, its compound forms $a|_{kl}^{ij}$, and the compound matrix $\underline{E_N}^{-1}|_{ij}^{12}$ are involved, hence the eigenfunction problem is said to be solved in an analytic way. Such an analytic form of calculation is useful especially for high frequencies and complex structures. One test has been made with Harkrider's oceanic model (Harkrider, 1970). The result is satisfactory even with frequency as high as 200 Hz.

If we start with different forms for ellipticity in equation (III-1-4), different formulas can be obtained following the same derivation procedures. They are

$$\begin{aligned}
\varepsilon &= -\frac{R_{12}}{R_{11}} &\rightarrow\quad \bar{B}_{m_i} &= \frac{X_{1j}\, Z|_{12}^{ji}}{R_{11}} \\
\varepsilon &= -\frac{R_{22}}{R_{21}} &\rightarrow\quad \bar{B}_{m_i} &= \frac{X_{2j}\, Z|_{12}^{ji}}{R_{21}} \\
\varepsilon &= -\frac{R|_{23}^{12}}{R|_{13}^{12}} &\rightarrow\quad \bar{B}_{m_{5-i}} &= \frac{(-1)^i\, Z_{ji}^{-1}\, R|_{j1}^{12}}{R|_{13}^{12}} \qquad \text{(III-1-8)} \\
\varepsilon &= -\frac{R|_{24}^{12}}{R|_{14}^{12}} &\rightarrow\quad \bar{B}_{m_{5-i}} &= \frac{(-1)^i\, Z_{ji}^{-1}\, R|_{j2}^{12}}{R|_{23}^{12}} \\
\varepsilon &= \frac{R^{-1}|_{34}^{13}}{R^{-1}|_{34}^{23}} &\rightarrow\quad \bar{B}_{m_{5-i}} &= \frac{(-1)^i X|_{ij}^{12} Z_{j2}}{R|_{23}^{12}} \quad .
\end{aligned}$$

Except for the third one, these formulas are not very useful. The reason for this statement becomes clear when a surface water layer is introduced in the next section.

Given the eigenfunctions at any depth, we can evaluate the energy integrals which are further used to find the quantities resulting from the perturbation of elastic properties. This technique is called the variational principle, which was introduced by Jeffreys (1961) and has been widely applied now (Takeuchi and Saito, 1972; Aki and Richards, 1980, chapter 7). The energy integrals needed to form the the system Lagrangian L_R for the Rayleigh wave,

$$L_R = \omega^2 I_0 - k^2 I_1 - 2k\, I_2 - I_3 \qquad \text{(III-1-9)}$$

are

$$I_0 = \int_0^\infty \rho\,[\,(\overline{U}_r^2) + (\overline{U}_z)^2\,]\,dz$$

$$I_1 = \int_0^\infty [\,(\lambda + 2\mu)(\overline{U}_r)^2 + \mu(\overline{U}_z)^2\,]\,dz$$

$$I_2 = \int_0^\infty [-\mu\overline{U}_z \frac{d\overline{U}_r}{dz} + \lambda\overline{U}_r \frac{d\overline{U}_z}{dz}\,]\,dz \qquad \text{(III-1-10)}$$

$$I_3 = \int_0^\infty [(\lambda + 2\mu)(\frac{d\overline{U}_z}{dz})^2 + \mu(\frac{d\overline{U}_r}{dz})^2\,]\,dz\ .$$

The Lagrangian is defined as the difference between the kinetic and potential energies. Its value is required to be zero if the system is at the eigen state. Hence, this function can be used to diagnose the numerical results. It should be noted that the eigenfunctions used in the integrals (equation III-1-10) are all normalized by the surface z-component displacement. The integrals in equation (III-1-10) can be expressed in terms of the integral I_{ij},

$$I_{ij} = \int \overline{B}_i \overline{B}_j \, dz$$

where

$$\overline{\mathrm{B}} = [\, U_r,\, U_z,\, T_z,\, T_r \,]^T \,/\, U_{z_1} \,.$$

The eigenfunction or motion-stress vector B_m in the m'th layer can be expressed as (equation II-1-14)

$$\mathrm{B}_m(z) = \mathrm{E}_m \,\Lambda(z)\, \mathrm{K}_m \,.$$

Taking the explicit forms of matrices Λ_m and K_m, we have

$$\mathrm{B}_m(z) = \mathrm{E}_m \cdot [\, A'' e^{\nu_\alpha z},\, B'' e^{\nu_\beta z},\, A' e^{-\nu_\alpha z},\, B' e^{-\nu_\beta z} \,]^T \,.$$

Since the dependence on depth z arises solely from the diagonal exponential matrix Λ, the integration of $B_i B_j$ over a given layer m with thickness d_m is easily found to be

$$\begin{aligned}
\int_{z_m}^{z_{m+1}} B_i\, B_j\, dz = {} & [\, E_{i1}E_{j1}\, (A'' e^{\nu_\alpha d})^2 + E_{i3}E_{j3}\, (A')^2 \,] \frac{1}{-2\nu_\alpha} (e^{-2\nu_\alpha d} - 1) \\
& + [\, E_{i2}E_{j2}\, (B'' e^{\nu_\alpha d})^2 + E_{i4}E_{j4}\, (B')^2 \,] \frac{1}{-2\nu_\beta} (e^{-2\nu_\beta d} - 1) \\
& + [\, (E_{i1}E_{j2} + E_{i2}E_{j1})\, (A'' e^{\nu_\alpha d})\, (B'' e^{\nu_\alpha d}) \\
& \quad + (E_{i3}E_{j4} + E_{i4}E_{j3})\, (A')\, (B') \,] \frac{1}{-(\nu_\alpha + \nu_\beta)} (e^{-(\nu_\alpha + \nu_\beta) d} - 1) \\
& + [\, (E_{i1}E_{j4} + E_{i4}E_{j1})\, (A'' e^{\nu_\alpha d})\, (B'')
\end{aligned} \qquad \text{(III-1-11)}$$

$$+ (E_{i2}E_{j3} + E_{i3}E_{j2})\,(B''e^{\nu_\beta d})\,(A')\,]\frac{1}{-(\nu_\alpha - \nu_\beta)}(e^{-\nu_\alpha d} - e^{-\nu_\beta d})$$

$$+ [\,(E_{i1}E_{j3} + E_{i3}E_{j1})\,(A''e^{\nu_\alpha d})\,(A')\,]\,d\;e^{-\nu_\alpha d}$$

$$+ [\,(E_{i2}E_{j4} + E_{i4}E_{j2})\,(B''e^{\nu_\beta d})\,(B')\,]\,d\;e^{-\nu_\beta d}\,,$$

where the layer index m has been omitted. In equation (III-1-11), A' and B' are part of elements of the vector K_m (or $\Lambda_m(0)\,\mathrm{K}_m$) which possess the values of potential constants at the top of layer, and $A''e^{\nu_\alpha d_m}$ and $B''e^{\nu_\beta d_m}$ are part of components of the vector $\Lambda_m(d_m)\,\mathrm{K}_m$ which is at the bottom of layer. These quantities can be determined from equation (II-1-15) and equation (II-1-21):

$$\mathrm{K}_m = [\,A''\,,B''\,,A'\,,B'\,]_m^T = \mathrm{E}_m^{-1}\,\mathrm{B}_m$$

$$\Lambda_m\;\mathrm{K}_m = [\,A''e^{\nu_\alpha d}\,,B''e^{\nu_\beta d}\,,A'e^{-\nu_\alpha d}\,,B'e^{-\nu_\beta d}\,]_m^T = \mathrm{E}_m^{-1}\,\mathrm{B}_{m+1}\,,$$

i.e., A' and B' are determined from eigenfunctions at the top of the layer and $A''e^{\nu_\alpha d_m}$ and $B''e^{\nu_\alpha d_m}$ from eigenfunctions at the bottom. If it were not done this way, the A', A'', B', and B'' could be very inaccurate by just using $\mathrm{E}_m^{-1}\,\mathrm{B}_m$. Suppose that the eigenfunctions have been determined precisely, the energy integrals as expressed in the analytic forms as equation (III-1-11) can be calculated with sufficient accuracy.

Equation (III-1-11) is applied to the layers lying above the half-space. In the half-space, the integral has the simpler form

$$\int_{z_N}^{\infty} B_i\, B_j\, dz = E_{i3}\, E_{j3}\, (A_N')^2 \frac{1}{2\nu_{\alpha_N}} + E_{i4}\, E_{j4}\, (B_N')^2 \frac{1}{2\nu_{\beta_N}} + (\, E_{i3}\, E_{j4} + E_{i4}\, E_{j3}\,)\, (A_N')\, (B_N') \frac{1}{\nu_{\alpha_N} + \nu_{\beta_N}}$$

where A_N' and B_N' are the elements of vector $\mathbf{K}_N$, and can be determined from $\mathbf{K}_N = \mathbf{E}_N^{-1}\, \mathbf{B}_N$.

Some of the integrals in equation (III-1-10) require the derivatives of U_r and U_z with respect to the depth. From the differential equation (II-1-11), these derivatives can be expressed in terms of the eigenfunctions U_r , U_z , T_z , and T_r as

$$\frac{dU_r}{dz} = kU_z - \frac{\omega^2}{\mu} T_r$$

$$\frac{dU_z}{dz} = \frac{1}{\lambda + 2\mu}(-k\,\lambda U_r + \omega^2 T_z\,)\;.$$

Therefore, the integrals in equation (III-1-10) as expressed in the form of I_{ij}'s are

$$I_0 = \sum_{n=1}^{N} \rho_n\, (I_{11} + I_{22})_n$$

$$I_1 = \sum_{n=1}^{N} (\lambda + 2\mu)_n (I_{11})_n + \mu_n\, (I_{22})_n$$

$$I_2 = \sum_{n=1}^{N} \omega^2[\sigma_n (I_{13})_n + (I_{24})_n] - k[(\lambda\sigma)_n (I_{11})_n + \mu_n (I_{22})_n]$$

$$I_3 = \sum_{n=1}^{N} k^2[\,(\lambda\sigma)_n (I_{11})_n + \mu_n (I_{22})_n\,] - 2k\,\omega^2[\sigma_n (I_{13})_n + (I_{24})_n] + \omega^4\,[(\sigma/\lambda)_n (I_{33})_n + (1/\mu)_n (I_{44})_n\,] \qquad \text{(III-1-12)}$$

where

$$\sigma = \frac{\lambda}{\lambda+2\mu} .$$

It is interesting to note that the Lagrangian L_R defined in equation (III-1-9) has the following form as expressed in I_{ij}'s:

$$L_R = \sum_{n=1}^{N} [(\rho-\frac{k^2}{\omega^2}\xi)_n (I_{11})_n + \rho_n (I_{22})_n - (\frac{\omega^2}{\lambda+2\mu})_n (I_{33})_n - (\frac{\omega^2}{\mu})_n (I_{44})_n]\omega^2 . \qquad \text{(III-1-13)}$$

The coefficients before the I_{ij}'s in this equation are just those of skew-diagonal elements in the differential equation (II-1-11). This property will be used in Appendix D for finding the amplitude factors from energy integrals.

<u>SH Eigenfunctions</u>

The above derivation was for the P-SV, or Rayleigh waves. We have found the analytic solutions of the eigenfunction at any depth, and the analytic forms for taking energy integrals. Similarly, for the Love waves we have the following properties:

(1) period equation $R_{66} = 0$.

(2) symmetry of Haskell matrices: $Z_{ij} = (-1)^{i+j} Z_{11-j,11-i}$ and $X_{ij}^{-1} = (-1)^{i+j} q_N' X_{11-j,11-i}$ with $q_N' = -2k^2\mu_N\nu_{\beta_N}/\rho_N^2$, and i,j=5,6.

(3) since $R_{5i} \cdot R_{i5}^{-1} = 1$, therefore $R_{66} \cdot R_{66}^{-1} = 1$

Now the eigenfunctions normalized with the surface tangential displacement become

$$\begin{aligned} \overline{B}_{m_k} &= Z_{k5} \\ &= X_{kp}^{-1} R_{p5} \\ &= X_{k6}^{-1} R_{65} \\ &= X_{k6}^{-1} / R_{56}^{-1} \\ &= \frac{(-1)^{k+1} X_{5,11-k}}{R_{56}} \qquad k=5,6 \ , \end{aligned} \qquad \text{(III-1-14)}$$

or equivalently

$$X_{5i} = (-1)^i R_{56} \overline{B}_{m_{11-i}} \qquad i=5,6 \ . \qquad \text{(III-1-15)}$$

The Lagrangian for Love waves is

$$L_L = \omega^2 I_0 - k^2 I_1 - I_2 \qquad \text{(III-1-16)}$$

with

$$\begin{aligned} I_0 &= \int_0^\infty \rho \, \overline{U}_\vartheta^2 \, dz \\ I_1 &= \int_0^\infty \mu \, \overline{U}_\vartheta^2 \, dz \\ I_2 &= \int_0^\infty \mu \, (\frac{d\overline{U}_\vartheta}{dz})^2 \, dz \ . \end{aligned}$$

The integration over one of the upper layers is

$$\int_{z_m}^{z_{m+1}} B_i \, B_j \, dz = [\, E_{i5} \, E_{j5} \, (C'' e^{\nu_\beta d})^2 + E_{i6} \, E_{j6} \, (C')^2 \,] \frac{1}{-2\nu_\beta} (e^{-2\nu_\beta d} - 1)$$

$$+ [\, (E_{i5} \, E_{j6} + E_{i6} \, E_{j5}) \, (C') \, (C'' e^{\nu_\beta d}) \,] \, d \, e^{-\nu_\beta d}$$

and over bottom half-space is

$$\int_{z_N}^{\infty} B_i B_j \, dz = E_{i6} \, E_{j6} \, (C')^2 \, \frac{1}{2\nu_\beta} \ .$$

The energy integrals as expressed in terms of I_{ij}'s take the forms:

$$
\begin{aligned}
I_0 &= \sum_{n=1}^{N} \rho_n \, (I_{55})_n \\
I_1 &= \sum_{n=1}^{N} \mu_n \, (I_{55})_n \qquad\qquad \text{(III-1-17)} \\
I_2 &= \sum_{n=1}^{N} \frac{1}{\mu_n} \, (I_{66})_n \quad .
\end{aligned}
$$

It is noted that there are some extra ω's present in equation (III-1-12) but not in equation (III-1-17). The reason arises from the somewhat arbitrary definitions for stress eigenfunctions which we chose in equation (II-1-7). The P-SV stresses used here should be multiplied by ω^2 before comparing with other formalisms such as that of Takeuchi and Saito (1972).

3.2 Normal Mode Theory

Normal mode theory is a method which uses the boundary value problem technique to deal with the waves propagating within layers. A normal mode defines a preferred frequency of vibration for the system. The surface wave, which is the most prominent phase on the seismogram, comes from the summation of the contributions from various preferred vibrations, or modes, of the system. There have been a number of investigations of this theory (Haskell, 1964; Ben-Menahem and Harkrider, 1964; Vlaar, 1966; Saito, 1967; and Levshin and

Yanson, 1971). Their results are principally equivalent (Tsai and Aki, 1970), although some expressional differences exist, especially for the source functions. In this section, a derivation will be performed using the results constructed in the last chapter, in which Saito's (1967) important extension to evaluate the residual contributions by means of variational principles will be intensively discussed.

The Fourier-Hankel transformed displacements at the free surface expressed in equation (II-2-8) are

$$\begin{aligned} U_{z_1}(\omega,k,n) &= S_i\, X|_{ij}^{12}\, Z_{j1} \,/\, R\,|_{12}^{12} & & \\ U_{r_1}(\omega,k,n) &= -S_i\, X|_{ij}^{12}\, Z_{j2} \,/\, R\,|_{12}^{12} & & i,j=1,4 \\ U_{\vartheta_1}(\omega,k,n) &= -S_j\, X_{5j} \,/\, R_{55} & & j=5,6 \;. \end{aligned}$$

From the discussion in section 2.2, the displacements at any depth for a double-couple dislocation source can be written as:

$$\begin{aligned} u_z(r,\vartheta,z,\omega) &= \int U_z^{(2)} J_2(kr)\,dk\cdot R_{ss} + \int U_z^{(1)} J_1(kr)\,dk\cdot R_{ds} + \int U_z^{(0)} J_0(kr)\,dk\cdot R_{dd} \\ u_r(r,\vartheta,z,\omega) &= \int -U_r^{(2)} J_1(kr)\,dk\cdot R_{ss} + \int -U_r^{(1)} J_0(kr)\,dk\cdot R_{ds} + \int U_r^{(0)} J_1(kr)\,dk\cdot R_{dd} \\ u_\vartheta(r,\vartheta,z,\omega) &= \int -U_\vartheta^{(2)} J_1(kr)\,dk\cdot R_{ss}' + \int -U_\vartheta^{(1)} J_0(kr)\,dk\cdot R_{ds}' \;, \end{aligned} \qquad \text{(III-2-1)}$$

where only the far-field terms are retained, and the R's describing the radiation patterns are given in equation (II-2-15). The superscripts representing the azimuthal mode number are parenthesized for clarity. Since the eigenfunctions found in the last section are all normalized with respect to U_{z_1} (P-SV) or U_{ϑ_1} (SH), the integral to be evaluated in equation (III-2-1) actually is

$$\int_0^\infty B^{(n)}(z)\, J_m(kr)\, dk = \int_0^\infty \bar{B}(z)\, U_{z_1}^{(n)}\, J_m(kr)\, dk$$

where $\bar{B}$ is the normalized eigenfunction at the depth z, and $U_{z_1}^{(n)}$ is the surface z-component displacement. Of course, for Love waves we use U_{ϑ_1} instead of U_{z_1}.

With the eigenfunctions available at different depths, we are ready to derive the surface wave fields. After laborious substitution and expansion, final results, similar to those of Saito (1967), Levshin and Yanson (1971), or Harvey (1981) will be found. Here we just show a component, RSS, for instance. With the aid of equation (III-1-7), the RSS component can be written as

$$\begin{aligned} RSS &= \int_0^\infty -\bar{U}_r(z)\, U_{z_1}^{(2)}\, J_1(kr)\, dk \\ &= \int_0^\infty -\bar{U}_r(z)\, \frac{S_i^{(2)}\, X\,|_{ij}^{12}\, Z_{j1}}{R\,|_{12}^{12}}\, J_1(kr)\, dk \\ &= \int_0^\infty -\bar{U}_r(z)\, (-1)^i\, \frac{R\,|_{13}^{12}}{R\,|_{12}^{12}}\, \bar{B}_{m_{5-i}}\, S_i^{(2)}\, J_1(kr)\, dk \end{aligned}$$

where the index m indicates the source layer. Setting the period equation $R\,|_{12}^{12} = 0$, and applying the residue theorem, we can find the surface wave displacements from the pole contributions:

$$RSS = -\pi i \sum [-\overline{U}_r(z)] \frac{R\,|^{12}_{13}}{\frac{\partial}{\partial k} R\,|^{12}_{12}} (-1)^l\, \overline{B}_{m_{5-l}}\, S_l^{(2)}\, H_1^{(2)}(kr) \;,$$

where the summation consists of each of the normal mode contribution at a given frequency. For far-field, i.e. r very large, the Hankel function $H_n^{(2)}(kr)$ has the approximation

$$H_n^{(2)} = \left[\frac{2}{\pi kr}\right]^{\frac{1}{2}} e^{-ikr+i\frac{\pi}{4}(1+2n)} \;.$$

By substituting the source function $4\pi\omega^2 S_4^{(2)} = -2k^2$, the RSS component surface wave at large distance becomes

$$RSS = \sum [-\overline{U}_r(z)]\, A_R \frac{1}{2\pi}\, k\, \overline{U}_{r_m} \left[\frac{2\pi}{kr}\right]^{\frac{1}{2}} e^{-ikr-i3\pi/4}$$

where

$$A_R = \frac{R\,|^{12}_{13}}{\frac{\partial}{\partial k} R\,|^{12}_{12}}\, \frac{k}{\omega^2} \;.$$

The amplitude factor or amplitude response A_R , defined by Harkrider (1964), can be evaluated in terms of phase velocity c, group velocity U, and the energy integral I_0 according to the following formula:

$$A_R = \frac{1}{2\, c\, U\, I_0} \qquad \text{(III-2-2)}$$

(Harkrider and Anderson , 1966; Levshin and Yanson , 1971; Takeuchi and Saito, 1972). The verification of this relation was first provided by Saito (1967). He employed the variational technique and decomposed the eigenfunction into two parts, one of which is continuous along the z-coordinate and the other is discontinuous by the interruption of the source at the source depth (Aki and Richards, 1980, p.310). However, such a splitting of the eigenfunction is not necessary if it is only desired to express the pole residue in terms of energy integrals. In Appendix D, we apply similar techniques from the variational principle, but use only the forms of the eigenfunction defined before. The derivation is relatively straightforward, but involved. An extra k and an extra ω^2, which were presented in section 2.3, automatically appear in our formulation.

Applying this result to the partial derivative of the period equation, we will obtain the excitation of surface waves in terms of the eigenfunction at the source depth and the energy integrals:

$$RSS = \sum [-\overline{U}_r(z)] \frac{1}{2cUI_0} \frac{1}{2\pi} k \, \overline{U}_{r_m} \left[\frac{2\pi}{kr}\right]^{\frac{1}{2}} e^{-ikr-i3\pi/4} .$$

The same procedures can be applied to all other components with the results:

$$u_z(r,\vartheta,z,\omega) = \sum D_{kR} \frac{\overline{U}_z(z)}{2cUI_0} \left[\frac{2\pi}{kr}\right]^{\frac{1}{2}} e^{-ikr-i\frac{\pi}{4}}$$

$$u_r(r,\vartheta,z,\omega) = \sum D_{kR} \frac{-\overline{U}_r(z)}{2cUI_0} \left[\frac{2\pi}{kr}\right]^{\frac{1}{2}} e^{-ikr-i\frac{3\pi}{4}} \qquad \text{(III-2-3)}$$

$$u_\vartheta(r,\vartheta,z,\omega) = \sum D_{kL} \frac{\overline{U}_\vartheta(z)}{2cUI_0} \left[\frac{2\pi}{kr}\right]^{\frac{1}{2}} e^{-ikr+i\frac{\pi}{4}} ,$$

where

$$D_{kR} = \frac{1}{2\pi}\left[(k\overline{U}_{r_m})R_{ss} + \{2\frac{k\,\alpha_m^2}{\rho_m}\overline{T}_{z_m} - k[3-(\frac{2\beta_m}{\alpha_m})^2]\,\overline{U}_{r_m}\}\,R_{dd} + i\,(\frac{k\,\beta_m^2}{\rho_m}\overline{T}_{r_m})\,R_{ds} \right]$$

$$D_{kL} = \frac{1}{2\pi}\left[k\,\overline{U}_{\vartheta_m}\,R_{ss}' - i\frac{1}{\mu_m}\overline{T}_{\vartheta_m}\,R_{ds}' \right] .$$

These solutions are the same as others (Saito, 1967; Herrmann, 1974; etc.). It is noted that at the free surface $\overline{U}_{r_1} = \varepsilon$ and $\overline{U}_{z_1}(0) = 1$. Hence the z- and r-component displacements at a particular frequency have the ratio $i\varepsilon$ (Haskell, 1953), i.e., a 90 degrees phase shift and ε maximum amplitude ratio.

Water Layer

If there is a water layer on the top of the solid layer stack, as in oceanic models, the situation does not become overly complicated. Only a simple modification of the formula derived above is needed. Denote the surface water layer by index '0'. The calculation of eigenfunctions in the water layer is just a case of acoustic wave propagation which is widely used in applied physics. The equations to be solved are

$$\nabla^2 \varphi = \frac{1}{\alpha^2}\frac{\partial^2\varphi}{\partial t^2}$$

$$u_r = \frac{\partial\varphi}{\partial r}$$

$$u_z = \frac{\partial\varphi}{\partial z}$$

$$T_z = \lambda\,\nabla^2\varphi \quad .$$

The boundary conditions become

$$\varphi = 0 \qquad at \quad z=0$$

$$\left.\begin{array}{l} u_{z_1} = u_{z_0} \\ T_{z_1} = T_{z_0} \end{array}\right. \qquad at \quad z=d_0$$

(Ewing et al, 1957, p.158). It is noticed that the u_r component is no longer continuous across the solid-liquid boundary, and $T_r = 0$ in the water layer, for which only compressional potential φ is still needed. The solutions similar to those of equation (II-1-4), (II-1-6), and (II-1-7) reduce to

$$\varphi(r,\vartheta,z,\omega) = \left.\begin{array}{c}\cos n\vartheta \\ \sin n\vartheta\end{array}\right\} J_n(kr)\Big\} \; Z_1(z)\Big\}$$

and $$Z_1(z) = A' e^{-\nu_\alpha z} + A'' e^{\nu_\alpha z} = -2A' \sinh \nu_\alpha z \;,$$

where we have used the free surface boundary condition, i.e., $Z_1(0) = 0$. The eigenfunctions are defined in the same way as before;

$$\begin{aligned} 4\pi\rho_0 u_r(r,\vartheta,z,\omega) &= \cos\vartheta\,[-\{-kZ_1^c\}\frac{dJ_n(kr)}{dkr}\,] + \sin\vartheta[\; c \;\rightarrow\; s \;] \\ 4\pi\rho_0 u_z(r,\vartheta,z,\omega) &= \cos\vartheta\,[\,\{\frac{dZ_1^c}{dz}\}\,J_n(kr)\,] + \sin\vartheta\,[\; c \;\rightarrow\; s \;] \qquad \text{(III-2-4)} \\ 4\pi T_z(r,\vartheta,z,\omega) &= \cos\vartheta\,[\,\omega^2\{-Z_1\}\,J_n(kr)\,] + \sin\vartheta\,[\; c \;\rightarrow\; s \;] \;, \end{aligned}$$

i.e.,

$$\begin{aligned} U_r &= -\frac{k}{\rho_0}Z_1 = \frac{2A'}{\rho_0}\,k\,\sinh\nu_\alpha z \\ U_z &= \frac{1}{\rho_0}\frac{dZ_1}{dz} = -\frac{2A'}{\rho_0}\nu_\alpha \cosh\nu_\alpha z \\ T_z &= -Z_1 = \quad 2A' \sinh\nu_\alpha z \quad . \end{aligned}$$

After normalization by U_{z_1} at $z = d_0$, we have

$$\begin{aligned}\overline{U}_r &= -k \sinh \nu_\alpha z \,/\, (\nu_\alpha \cosh \nu_\alpha d_0) \\ \overline{U}_z &= \cosh \nu_\alpha z \,/\, \cosh \nu_\alpha d_0 \\ \overline{T}_z &= -\rho_0 \sinh \nu_\alpha z \,/\, (\nu_\alpha \cosh \nu_\alpha d_0) \quad .\end{aligned} \tag{III-2-5}$$

Since there is the same Bessel function dependence for T_z and U_z components as in equation (III-2-4), we find that

$$\overline{T}_{z_1} = (-\frac{\rho_0 \sinh \nu_\alpha d_0}{\nu_\alpha \cosh \nu_\alpha d_0})\, \overline{U}_{z_1} = T\, \overline{U}_{z_1}$$

at the solid-liquid boundary. T was also derived by Harkrider (1964) from the ratio a_{32}/a_{22} for the water layer $\beta_0 = 0$.

Now let us check the effect of the water layer on the results derived before. The surface displacements to be found are now at the top of the solid layers. First, the period equation is altered to appear as

$$\begin{bmatrix} 0 \\ 0 \\ A_N' \\ B_N' \end{bmatrix} = \mathrm{R}\cdot \begin{bmatrix} U_{r_1} \\ U_{z_1} \\ T_{z_1} \\ 0 \end{bmatrix} = \mathrm{R}\cdot \begin{bmatrix} U_{r_1} \\ U_{z_1} \\ TU_{z_1} \\ 0 \end{bmatrix} , \tag{III-2-6}$$

where R includes the matrices for solid layers only. The first two rows give the new period equation:

$$R\,|_{12}^{12} + T\,R\,|_{13}^{12} = 0 \quad .$$

It is noted that because the period equation changes, so do the dispersion values. The form of the ellipticity relation at the solid-liquid boundary is not affected if we use the following derivation:

$$\varepsilon = \frac{U_{r_1}}{U_{z_1}} = -\frac{R_{12} + TR_{13}}{R_{11}} = -\frac{R_{22} + TR_{23}}{R_{21}}$$

$$= -\frac{R_{12}R_{23} + TR_{13}R_{23}}{R_{11}R_{23}} = -\frac{R_{22}R_{13} + TR_{23}R_{13}}{R_{21}R_{13}}$$

$$= -\frac{R_{12}R_{23} - R_{22}R_{13} + T(R_{13}R_{23} - R_{23}R_{13})}{R_{11}R_{23} - R_{21}R_{13}} = -\frac{R\,|^{12}_{23}}{R\,|^{12}_{13}} \,, \qquad \text{(III-2-7)}$$

or equivalently,

$$\varepsilon = \frac{R^{-1}|^{14}_{34}}{R^{-1}|^{24}_{34}} \,. \qquad \text{(III-2-8)}$$

However, other forms, such as those in equation (III-1-4), can no longer be used. For example, there is yet another form:

$$\varepsilon = -\frac{R\,|^{12}_{24} + T\;R\,|^{12}_{34}}{R\,|^{12}_{14}} \,.$$

As a consequence, the eigenfunction formula (equations III-1-6 and III-1-8), if expressed in the corresponding forms with equations(III-2-7) and (III-2-8), will stay the same, but this is not true for the other forms. This point should be noticed when a water layer is imposed.

The energy trapped in the water layer is easy to obtain by direct integration from equation (III-2-5). Here we just list the results needed to calculate the Lagrangian:

$$\int_0^{d_0} \overline{U}_r \overline{U}_r \; dz = \frac{k^2}{2A\,\nu_\alpha^2}\,[\,B - d_0\,]$$

$$\int_0^{d_0} \overline{U}_z \overline{U}_z \, dz = \frac{1}{2A} [B + d_0]$$

$$\int_0^{d_0} \overline{U}_r \frac{d\overline{U}_z}{dz} \, dz = \frac{-k}{2A} [B - d_0]$$

$$\int_0^{d_0} \frac{d\overline{U}_z}{dz} \frac{d\overline{U}_z}{dz} \, dz = \frac{\nu_\alpha^2}{2A} [B - d_0]$$

where

$$A = \cosh^2 \nu_\alpha d_0 \qquad B = \frac{\sinh 2\nu_\alpha d_0}{2\nu_\alpha} \quad .$$

The Lagrangian integral components I_i have the following forms, obtained by setting $\mu_0 = 0$ in equation (III-1-10),

$$I_0 = \int \rho_0 (\overline{U}_r^2 + \overline{U}_z^2) \, dz$$

$$I_1 = \int \lambda_0 \overline{U}_r^2 \, dz$$

$$I_2 = \int \lambda_0 \overline{U}_r \frac{d\overline{U}_z}{dz} \, dz$$

$$I_3 = \int \lambda_0 \frac{d\overline{U}_z}{dz} \frac{d\overline{U}_z}{dz} \, dz \quad .$$

These integrals over the water layer should be added to the corresponding integrals for solid layers derived in section 3.1. Although the displacement fields calculated are at the top of solid layers, we find part of the wave energy is trapped in the first water layer, especially for high frequency signals. When using the energy viewpoint to approach the eigenfunction problem, the contribution from the water layer cannot be simply ignored. The presence of the water layer does not affect the corresponding SH solution.

Synthetic High Frequency Seismograms

In the above discussion, we have set up a complete system to calculate eigenfunctions and the related parameters. A FORTRAN program was designed to compute the surface wave time history or its spectrum for relatively high frequencies and moderately complex models. An explanation of this program is given in Appendix E.

Figures 12, 13, and 14 show examples obtained from these programs for three different components. In each figure the upper five seismograms are obtained by using a symmetric triangular source time function with base of one second, and the bottom seismogram from a step source time function. The frequencies cover the range from 0 to 10 Hz, although the triangular source has the corner frequency at 0.7 Hz. The CUS model in Table 1 is used. Source parameters are dip = 50°, slip = 180°, strike = 40°, and depth = 14 km. The receivers are located to the north of source, and the epicentral distances are indicated at the end of each seismogram. Furthermore, a Q-model with $Q_\beta = 250$ for the top 24 km depth followed by $Q_\beta = 2000$ for the halfspace is also assumed. In these figures an apparent low frequency fundamental mode signal arives after some high frequency Lg (Airy) phases. These synthetic seismograms seem to be of quite high quality. Even the step-source generated seismograms, which contain 10 Hz frequen-

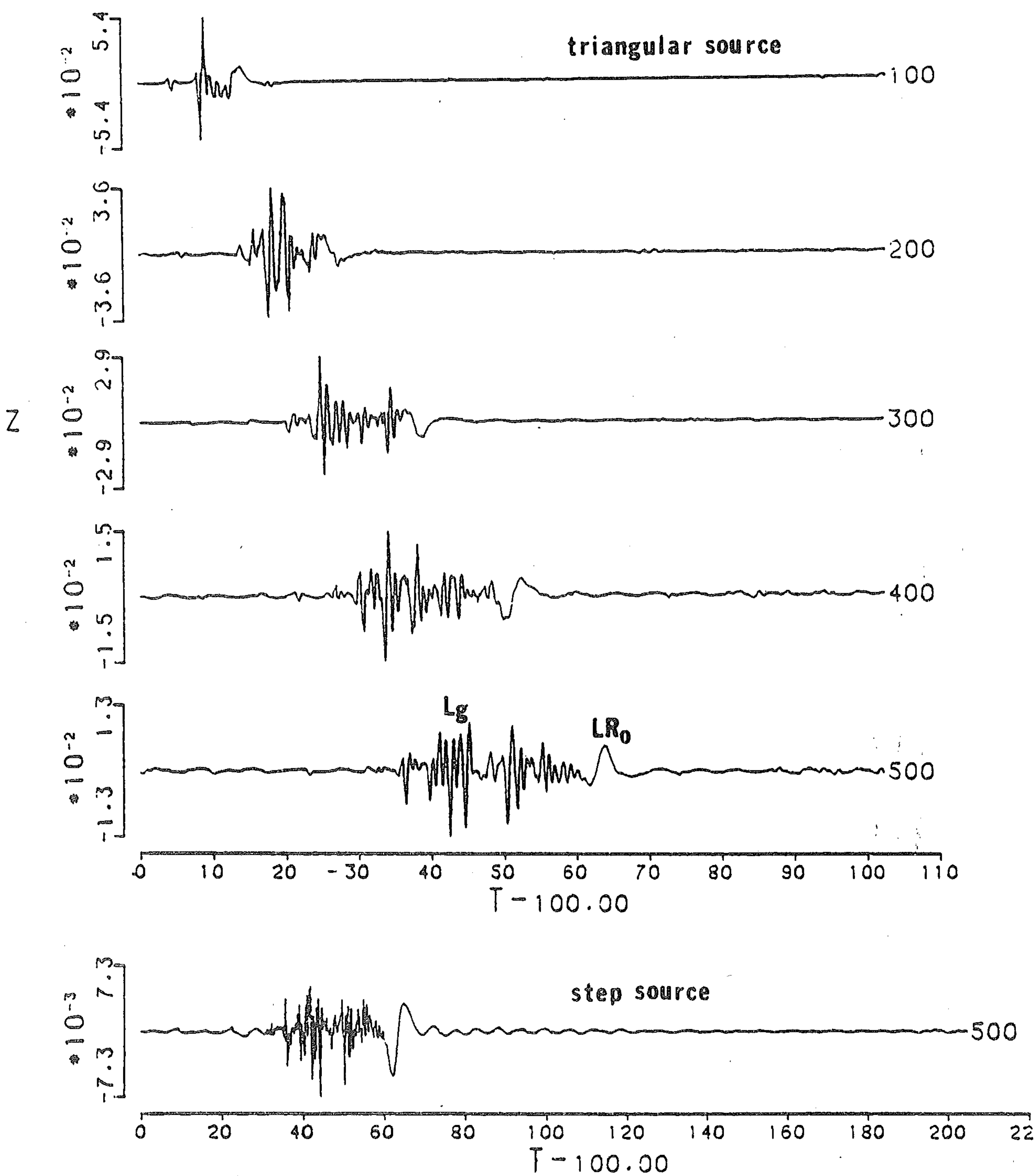


Figure 12. Theoretical seismograms generated by eigenfunction programs. The upper five seismograms are due to a dislocation source with a triangular source time function buried at the depth of 14 km in CUS model. The frequencies used cover the range from 0 to 10 Hz. The bottom seismogram is due to the same dislocation source but with a step source time function. A Q-model with $Q_\beta = 250$ for top 24 km and $Q_\beta = 2000$ for other layers is used. The number at the end of each seismogram indicates the epicentral distance.

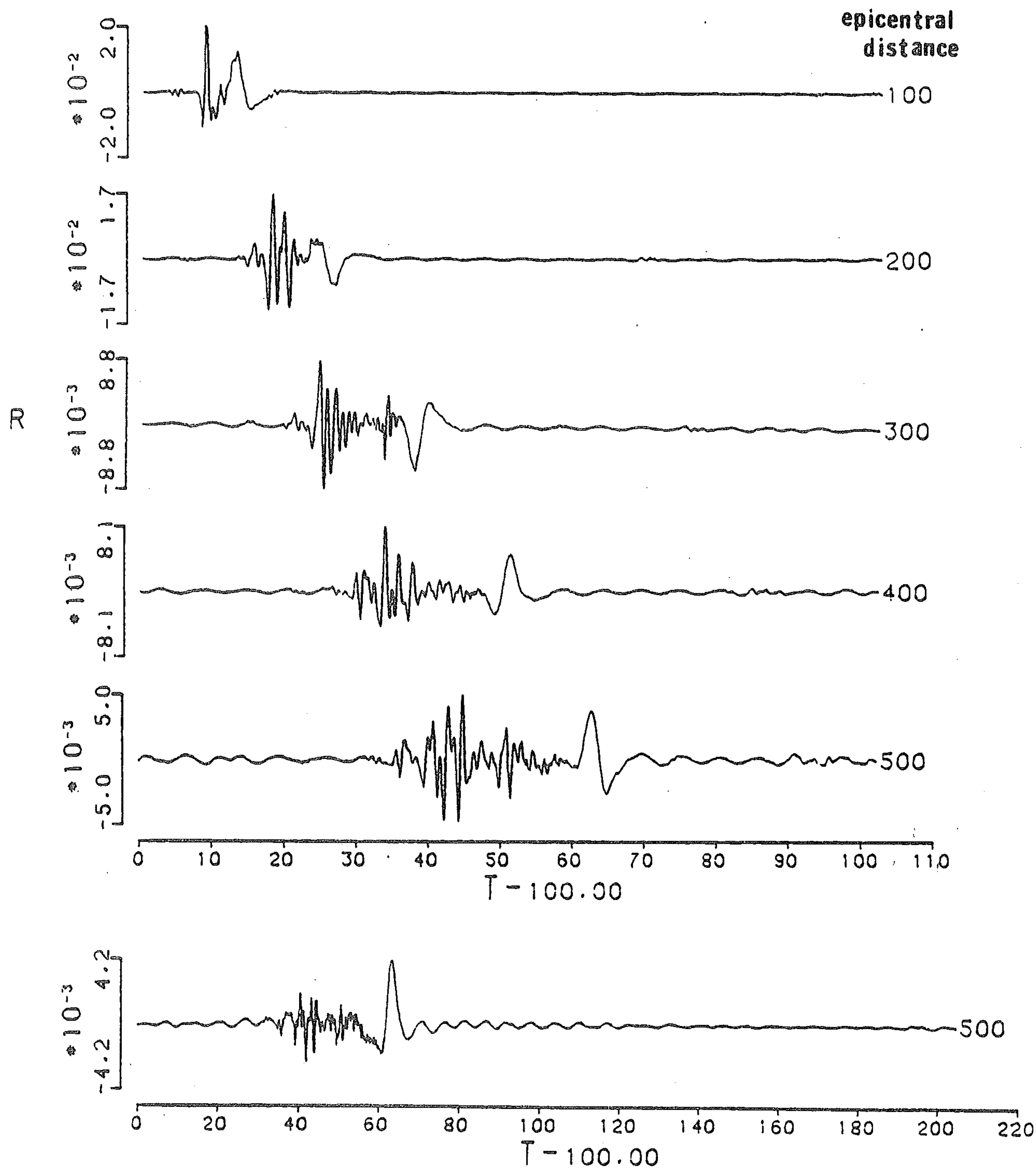


Figure 13. Results corresponding to Figure 12 but for the radial component.

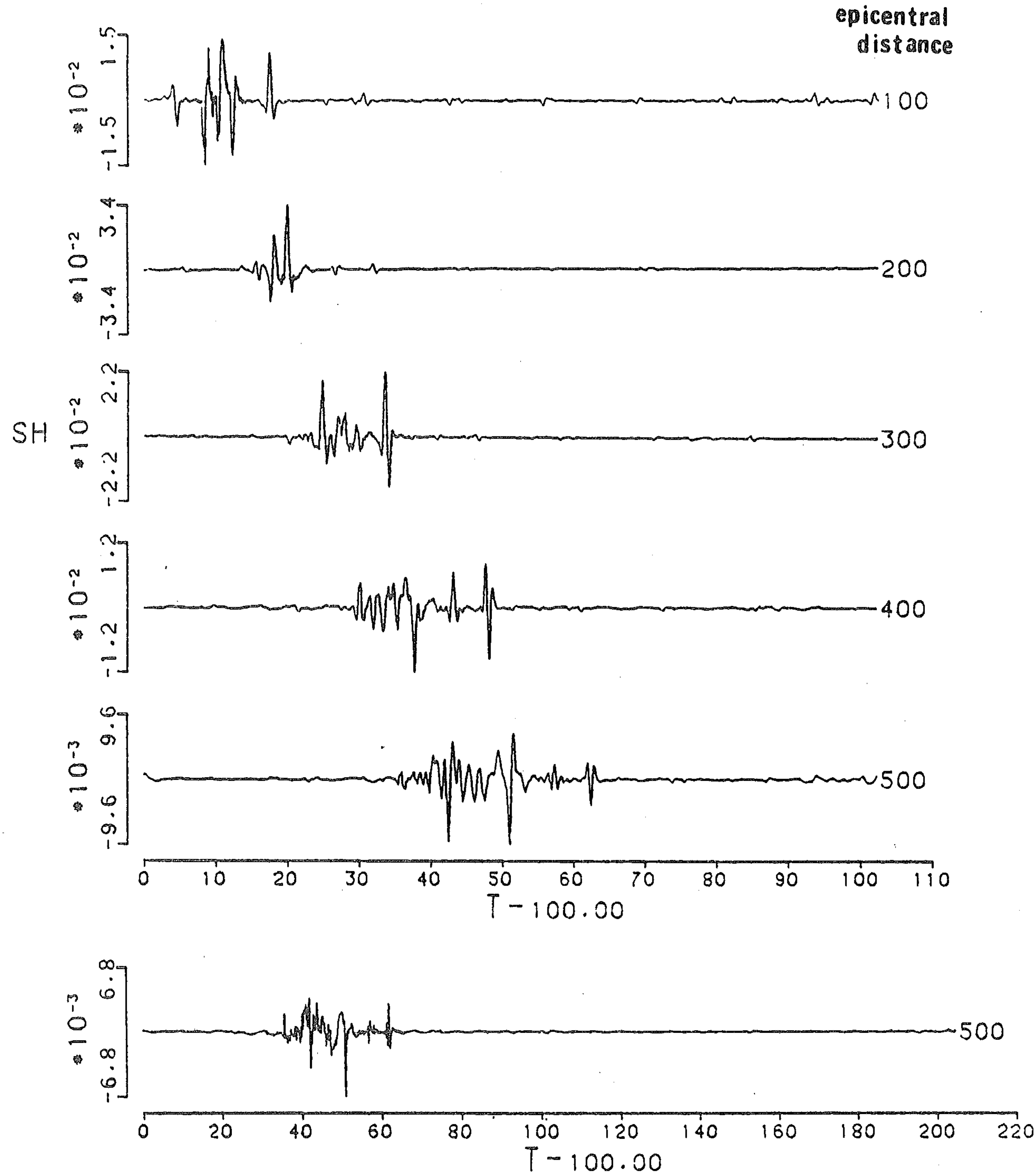


Figure 14. Results corresponding to Figure 12 but for the tangential component.

cies, exhibit well developed and fairly noiseless waveforms.

CHAPTER IV

BODY WAVE - LEAKY MODE STUDY

In the previous chapter, the normal mode theory of surface waves was considered. The objective of this chapter is to present a detailed discussion about the generation of body waves. This part of the waveform comes from, at least for most P-SV cases, the branch line integral discussed in section 2.4. Figure 15a shows the contributions of the pole residue (middle), branch line integral (bottom), and the total seismogram (top) of the REP component with source depth = 10 km and r = 100 km for the SCM model of Table 1. Figure 15b gives a similar display for the RDS component with source depth = 1 km and r = 25 km for the CUS model. It is obvious that there are two definitely different signals arising from the pole contribution and the branch line integral, which constitute the final seismogram. This fact can further be seen from Figures 16 and 17. However for the SH case, the situation is totally different. Figure 18 indicates that, for the SH case, the pole contribution constitutes most of the signal, and the branch line integral is only required to make the total seismogram 'causal' (Herrmann, 1978a). Thus, what is the role of the branch line integral in constructing a seismogram? This question will be discussed in this chapter using the leaky mode

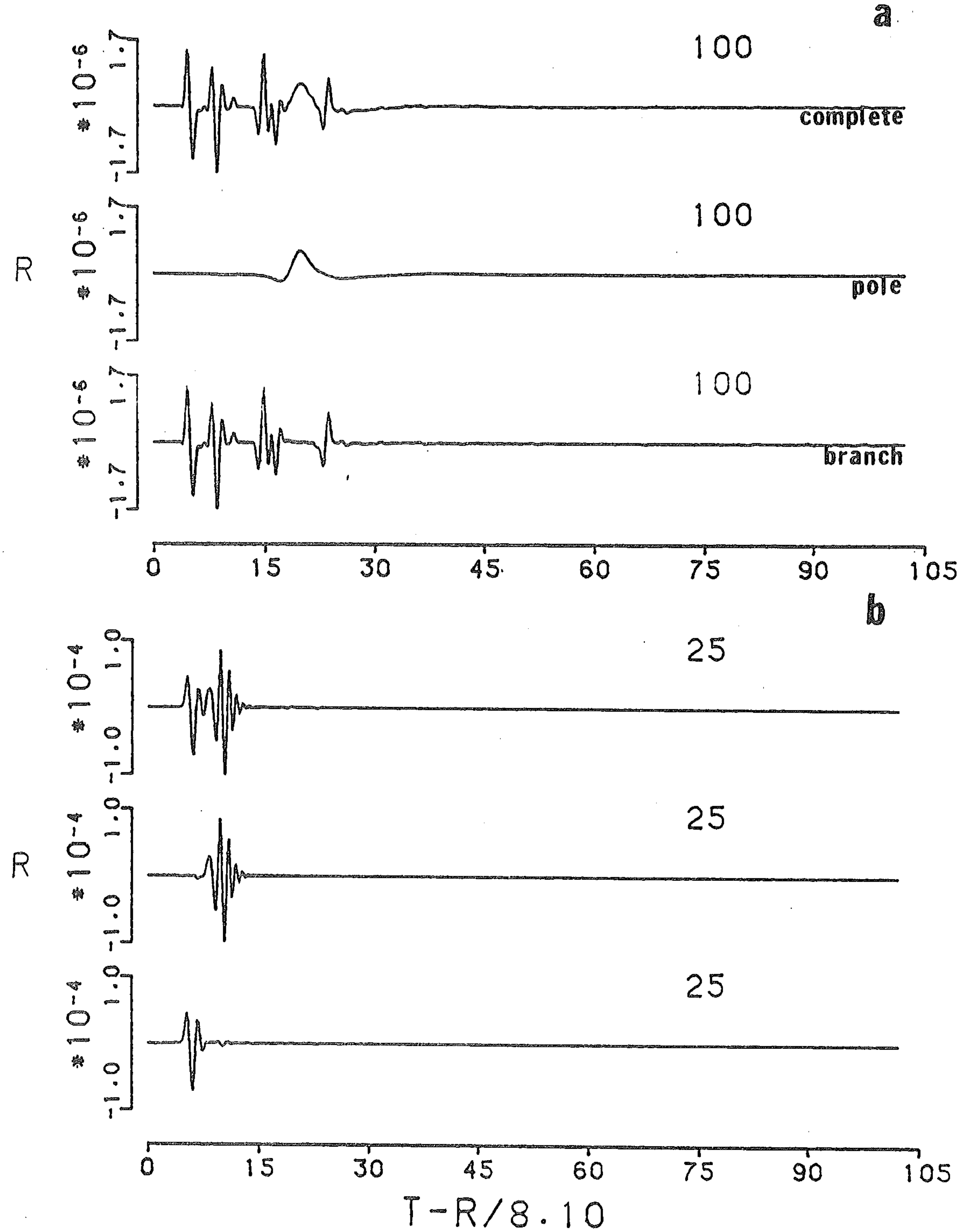


Figure 15. Study of contribution of various components of contour integration. (a) is the set of vertical component seismograms due to an explosive source at 100 km away and buried at 10 km depth in SCM model. (b) is the set of radial component seismograms due to a 45° dip-slip source at 25 km away and buried at 1 km depth in CUS model. In each set of seismograms, the top one is the complete solution, the middle is the pole contribution, and the bottom is the contribution from branch line integral.

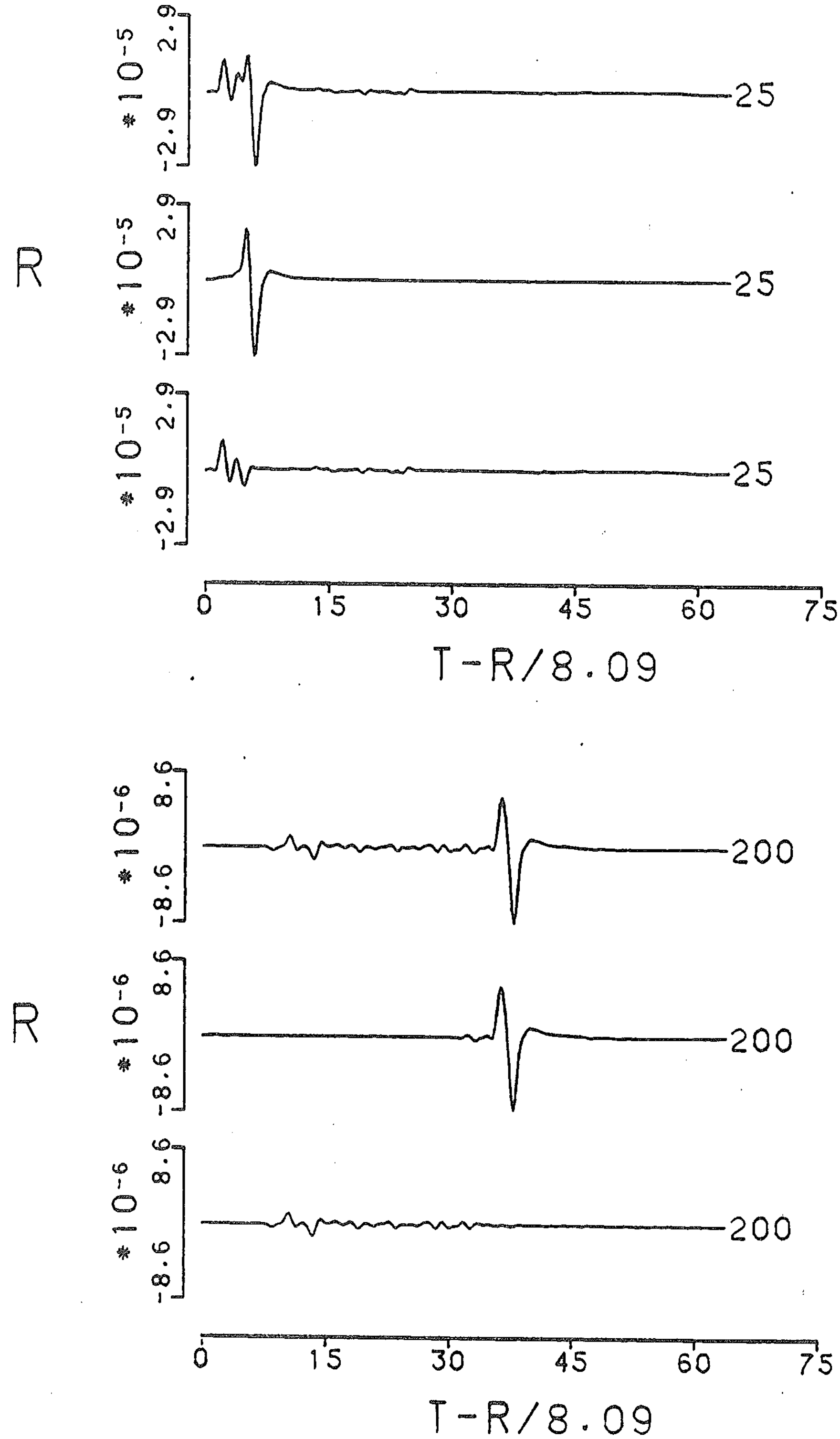


Figure 16. Same comparison as for Figure 15. The radial component seismograms due to a dip-slip source buried at 10 km depth in SCM model are displayed. Two sets of seismograms correspond to epicentral distances at 25 km and 200 km, respectively.

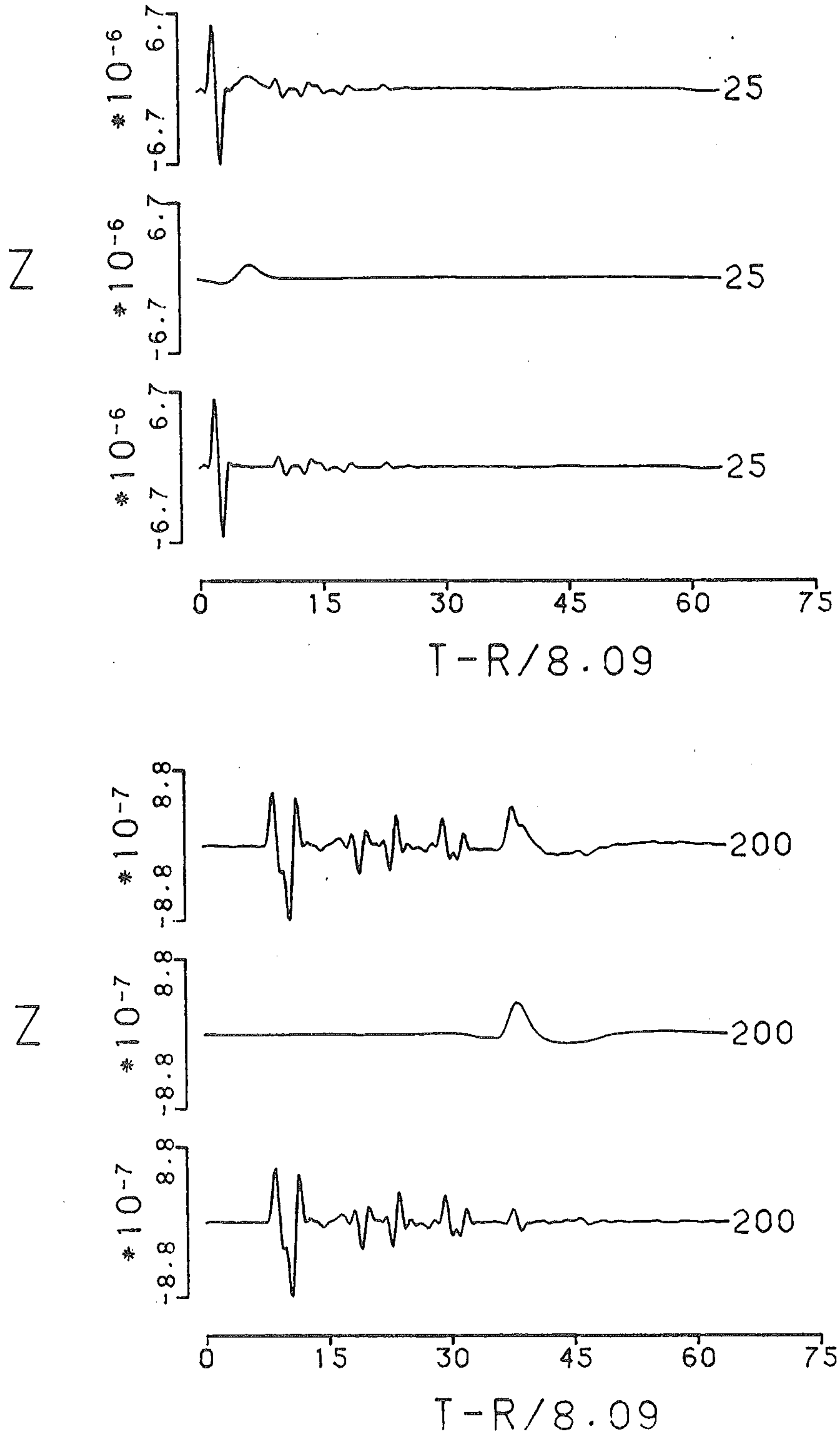


Figure 17. Same comparison as Figure 16, but for the vertical component.

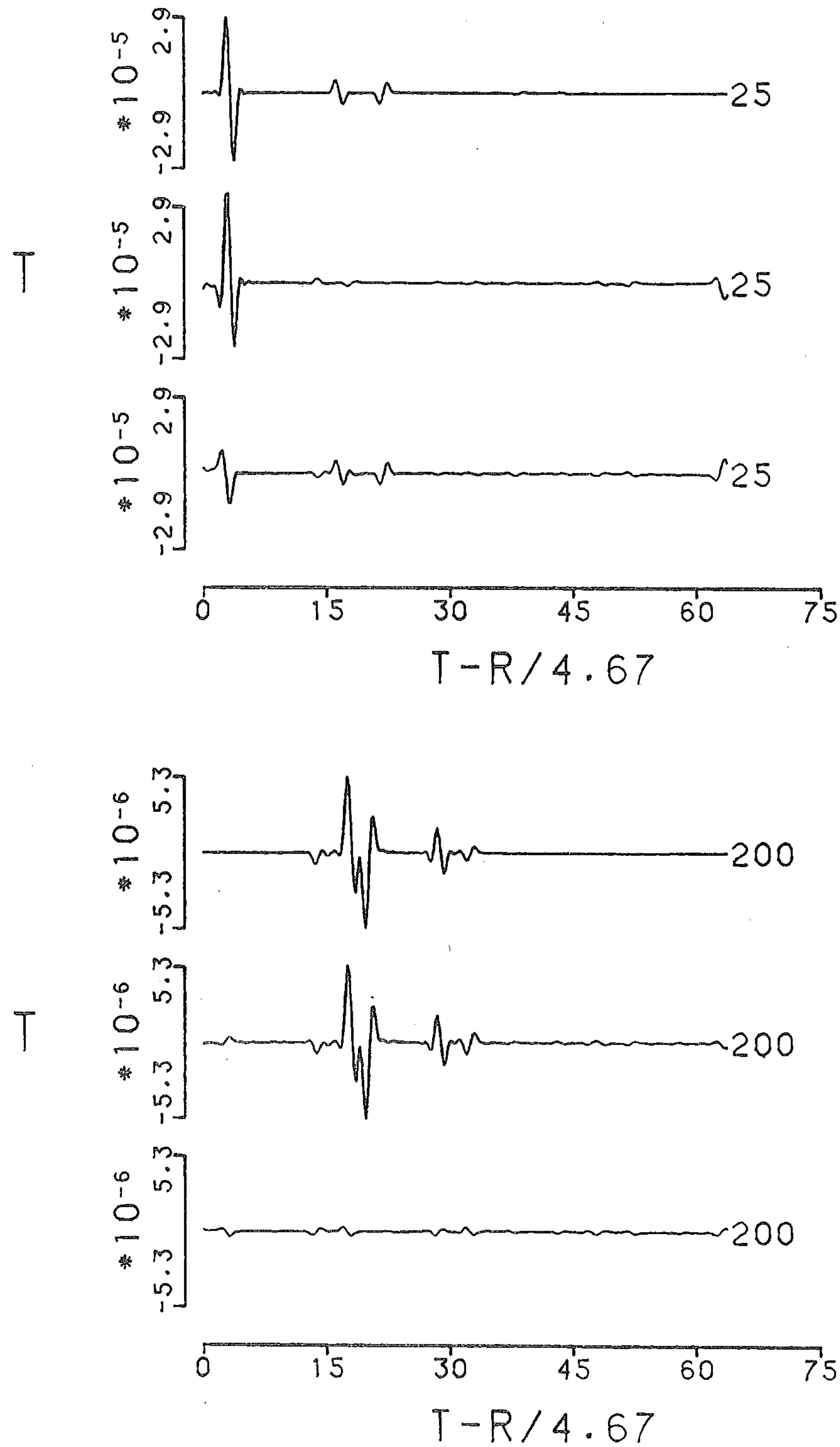


Figure 18. Same comparison as Figure 16, but for the tangential component.

approach.

Harvey (1981) imposed a deep and rigid cap at the bottom of the structure to lock the energy in the upper layers. It was found that this process is an approximation which ignores the leakage of waves and forces the leaky mode to appear on the real wavenumber axis as pseudo-normal modes. Since the normal modes are usually easy to handle, this approach provides a good way to simulate the body waves.

The reflectivity method of Fuchs and Müller (1971) is a widely used method for calculating body waves. Using a theory from Kennett (1974), we are able to generalize the reflectivity method. Most importantly, such a modification is made by simply adjusting our system constructed in chapter II. Hence, the reflectivity method is only a special case of our system.

4.1 The Influence of Leaky Modes on Body Waves

In section 2.4, we constructed the full wave field after combining two different types of solutions: the pole contribution and the branch line integral. The poles which describe the vibration modes of layered wave guides exhibit dispersion and contribute mainly to the surface wave. This point was further confirmed by the normal mode approach as presented in chapter III.

It is natural to ask whether the branch cut integral naturally gives rise to the body wave. In this section, we will extend the normal mode theory to include the leaky mode (or leaking mode), and investigate the role of branch line integration by means of the leaky mode approach.

The study of leaky modes began when the computer was still in its infancy. The direct evaluation of a line integral along the branch cuts is difficult. To overcome this, two methods were proposed. One was the use of the steepest descent method to find the approximate solutions for large source-receiver distances (Lapwood, 1949; Fuchs, 1971). The other was an attempt to deform the integration path into other Riemann sheets and evaluate the pole residue there (Rosenbaum, 1960; Phinney, 1961). The branch cut and Riemann sheets were introduced in order to make the vertical wavenumber single valued. There exist four sheets for P-SV with respect to four combinations of Re(ν_α), Re(ν_β) being positive or negative, and two sheets for SH. We will use (+,-) for positive Re(ν_α) and negative Re(ν_β), respectively, and similar sign symbols for other combinations. The poles in the lower Riemann sheets are called leaky modes, as distinct from normal modes. Since these poles are situated off the real axis in the complex plane, imaginary parts cause the attenuation as waves propagate with distance or time.

Physically, the leaky modes describe traveling disturbances as a leaking S wave or P and S waves, respectively, into the substratum. At moderate distances, such a mode might be seen as it sometimes results in large amplitude dispersed oscillatory trains appearing in the interval between the P and S arrivals, which are called PL waves.

Many authors have explored leaky modes by extending the normal (or trapped) mode theory (Phinney, 1961; Gilbert, 1964; Laster et al, 1965; Haskell, 1966; Abramovici, 1968; Alsop, 1970; Cochran et al, 1970; Dainty, 1971; Watson, 1972). Among these works, that of Gilbert (1964) is of fundamental importance. Using real k-complex f as independent variables, Gilbert classified the modes into two types: Lamb's roots and organ pipe roots, when k approaches zero. Lamb's roots are associated largely with the properties of the half-space. One of these roots is the fundamental mode of the Rayleigh wave which occupies (+,+) position over the whole range of k-f. But the other Lamb's roots behave like organ pipe modes as k or f become large. An organ pipe mode represents the standing wave trapped in the layer. It is the mode associated with the energy reverberated almost vertically inside the layers. Two forms of organ pipe mode exist: π modes which possess P-wave properties and originate from the (-,+) sheet, and Σ modes which have S-wave properties and come from the (+,-) sheet.

Following the work of Gilbert (1964), investigation principally concentrated on the search for modes, i.e., defining the dispersion curve and its attenuation relation. Laster et al (1965) provided a theoretical basis for the dispersion study. They also generated theoretical seismograms from leaky mode contributions and indicated the importance of the leaky mode in the early part of the seismogram by comparison to experimental data. Cochran et al (1970) displayed the dispersion curves for π and Σ modes in a multiple elastic wave guide, and found lattice dispersion patterns which are characterized by π-pseudo and Σ-pseudo modes. The π-pseudo modes were shown to make up the oscillatory part of the seismogram between P and S, which depends solely on the P velocities. Abramovici (1968) extended the compound matrix technique to the leaky mode calculation, and defined a transfer function to study the contribution of individual modes. Except for Laster et al (1965), the real k-complex f approach of Gilbert (1964) has been employed to compute dispersion curves by all other investigators. Watson (1972) made an interesting study using real f-complex k analysis. The most significant contribution of his work was the identification of PL and OP modes. His approach is especially suitable for our present study. It will be seen that it is these two modes which affect the response along the branch cut and enter into the final contribution of seismogram synthesis.

Following Watson (1972) we take the frequency to be real and allow the poles to wander on the complex wavenumber plane. The branch cuts are kept along the real and imaginary axes, as before. This differs from the cuts used by other studies (Gilbert, 1964;, Laster et al , 1965; Watson, 1972), in which the branch cuts were usually made directly in the first or fourth quadrant to expose the lower Riemann surfaces. Our object is to investigate the influence of leaky mode poles on the response function along the real branch cut.

It is already known (Gilbert, 1964) that the poles in the complex plane occur in sets of four, that is, if k is a pole, so are $-k$, k^* , and $-k^*$. Here we will restrict ourselves to the fourth quadrant only. Keeping the branch cut along the real and imaginary axis, the radical ν_α has values in the fourth quadrant given by

$$\nu_\alpha = \pm\left[\sqrt{\frac{r+x}{2}} - i\sqrt{\frac{r-x}{2}} \right] \qquad \text{(IV-1-1)}$$

where

$$x = k_R^2 - k_I^2 - k_\alpha^2$$
$$y = 2\, k_R\, k_I$$
$$r = \sqrt{x^2 + y^2} \quad .$$

A similar definition for ν_β can be obtained by changing α to β .

Poles were searched for on the lower complex-k sheets corresponding to different frequencies. When frequency changes, the leaky poles also transit along some paths, and sometimes even pass across the branch cut and appear on other Riemann sheets. Since our branch line integrations are carried out on the (+,+) sheet, we will consider only those poles which have the ability to enter this top sheet. When poles transit through the branch cut between $k_{\alpha N}$ and $k_{\beta N}$, only ν_α changes sign, but through the cut between Ø and $k_{\alpha N}$ both signs of the radicals change. Hence the candidates are those poles just below the $k_{\alpha N} - k_{\beta N}$ cut on the (+,-) sheet and those near the cut between the origin and $k_{\alpha N}$ on the (-,-) sheet. These two regions are called region I and region II by Watson (1972) and Pilant (1979). The poles on the (-,+) sheet have no effect on the integration, however they might transit into (+,-) or (-,-) sheets and become important.

A computer program was set up to find the curves in a finite region of interest in the complex-k plane, such that the real or imaginary parts of the period equation are zero. The intersections of these curves are taken as the roots of the leaky modes. Since we are only interested in their properties, no refinement of pole position is necessary. Figure 19 shows the curves of the null real part of the period equation (denoted by '+' sign), and curves with the null imaginary part (denoted by 'x') at frequencies 0.32 Hz and 0.33 Hz. The model used is the SCM model listed in Table 1. These curves exhibit a particular pattern, which will help us to identify a specified pole as the frequency varies. The pattern obviously shows that two kinds of poles exist, the OP modes which always remain on the (+,-) sheet, follow an exponential type of path from $-i\infty$, and the PL modes which wander in the vicinity of the $k_{\alpha N} - k_{\beta N}$ branch cut.

Using these, we made a close search for poles by varying the frequency from 0.25 Hz to 0.40 Hz in steps of 0.01 Hz. The results are shown in Figures 20 to 22. Now the roles of the OP and PL modes are clear. In Figure 20 the OP poles migrate from the region with large imaginary k, i.e. high attenuation, and approach the point k_{α_1}. The PL modes, on the other hand, emerge from the (-,+) sheet by crossing the cut around k_{α_1}, and shift slowly in the region very close to the branch cut. These poles will contribute to the integration along the cut, which under some particular conditions are significant enough to generate a dispersive wavetrain between P and S, called PL waves. The PL and OP modes collide at the place just beneath k_{α_1}. After this, two kinds of modes mingle together and generate

(+, -)

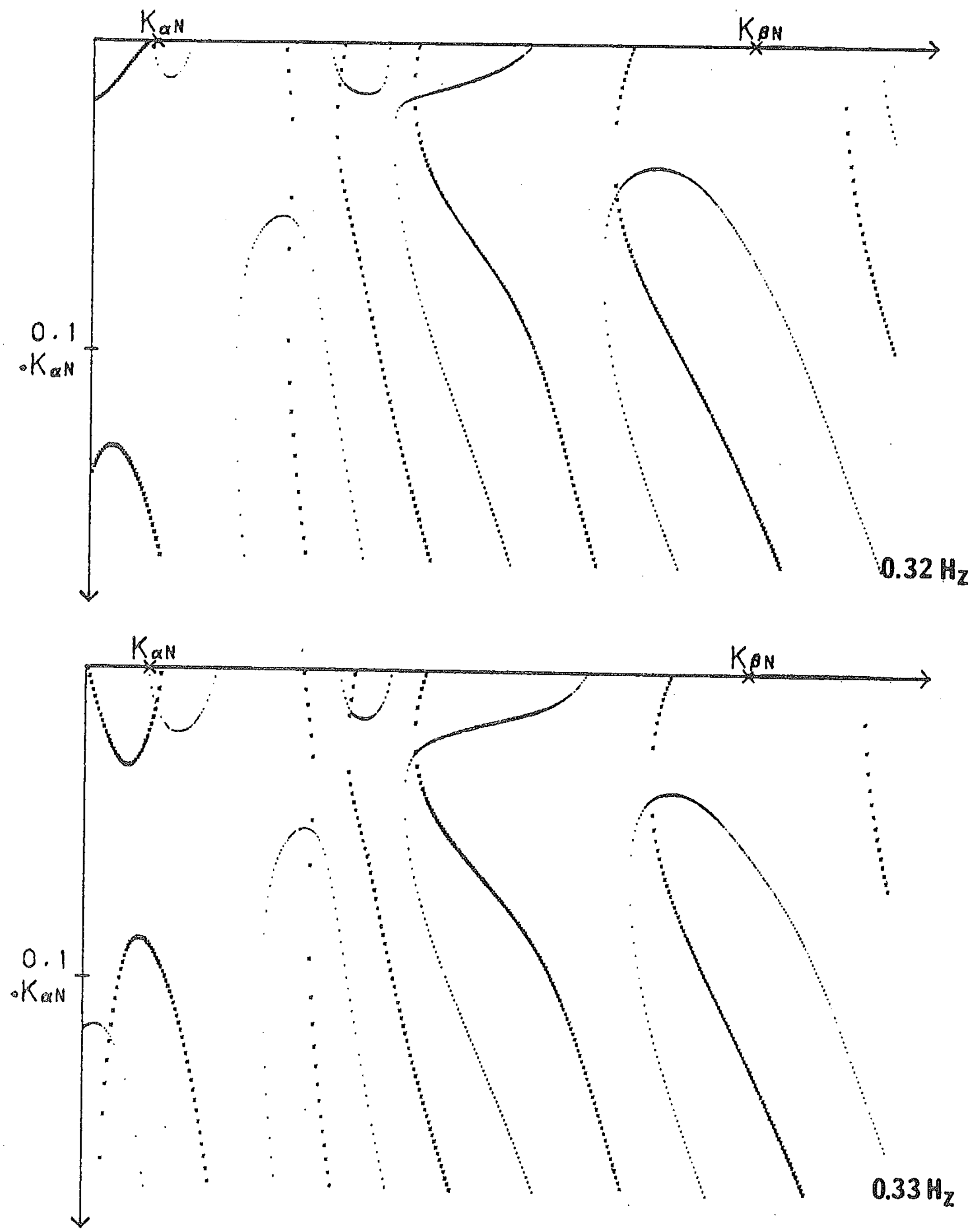


Figure 19. Curves of the null real part of Rayleigh wave period equation (denoted by '+' sign) and the null imaginary part (denoted by 'x' sign) in the fourth quadrant of (+,-) sheet of complex k plane. k_{α_N} and k_{β_N} are branch points. The top figure is obtained at 0.32 Hz and the bottom at 0.33 Hz. The SCM model is used.

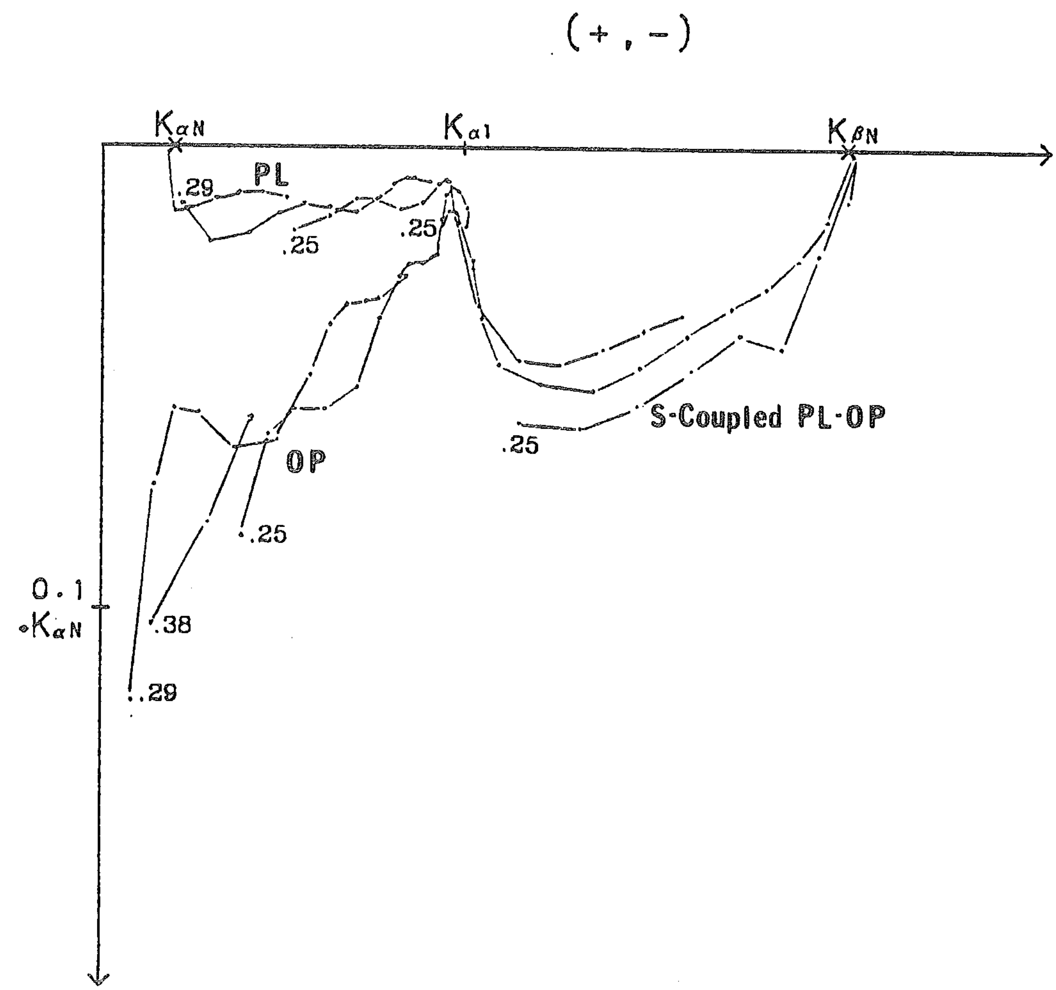


Figure 20. Paths of leaky modes of SCM model through the fourth quadrant of (+,-) sheet of complex k plane. The frequencies change from 0.25 Hz to 0.40 Hz. Two kinds of modes, namely PL and OP modes, exist before k_{α_1} the wavenumber corresponding to the first layer P velocity. After this point, two modes mingle together and form the shear-coupled PL-OP mode. The numbers at the beginning of each path indicate the starting frequencies.

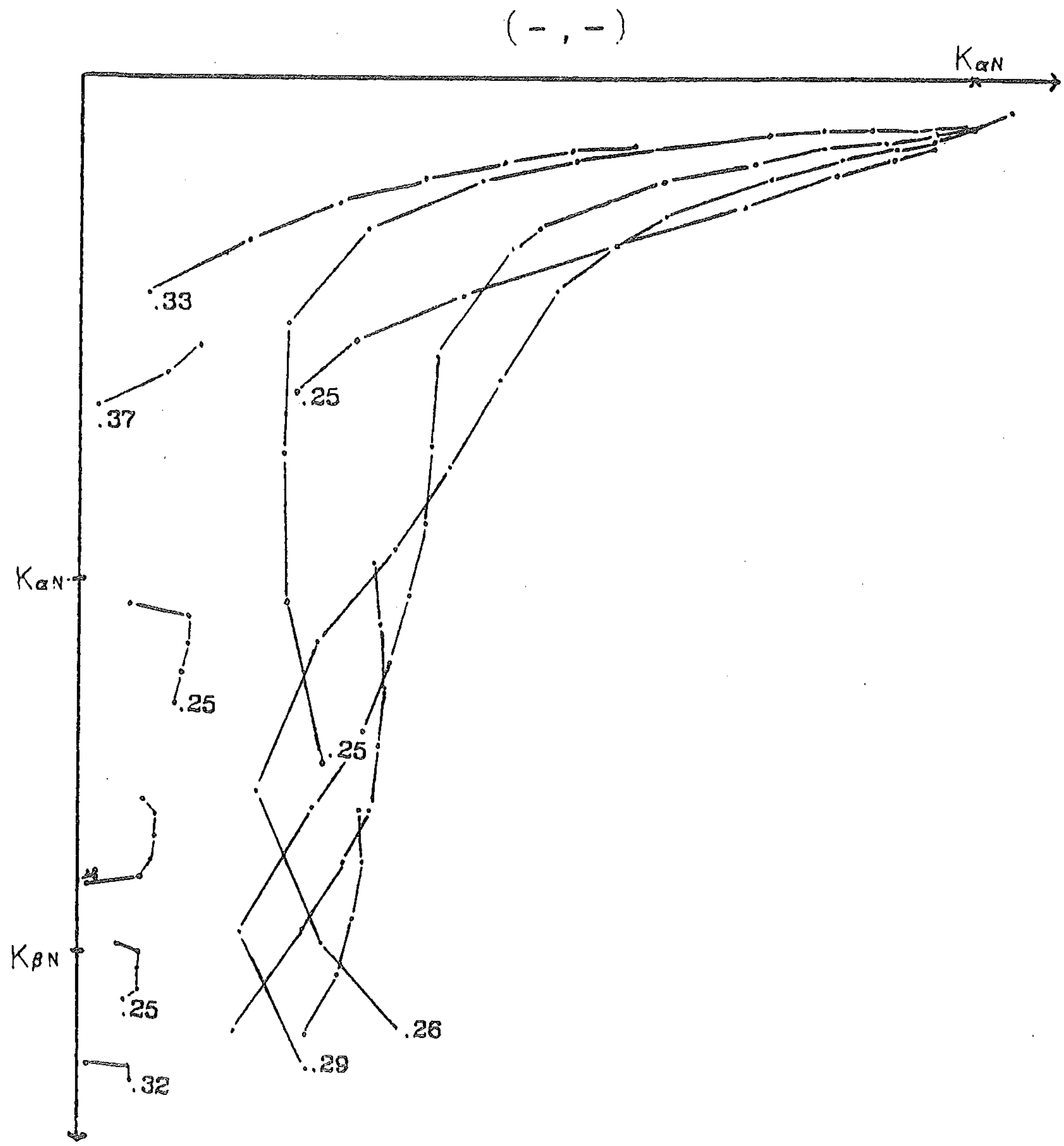


Figure 21. Same as Figure 20, but for leaky modes on (-,-) sheet of complex k plane.

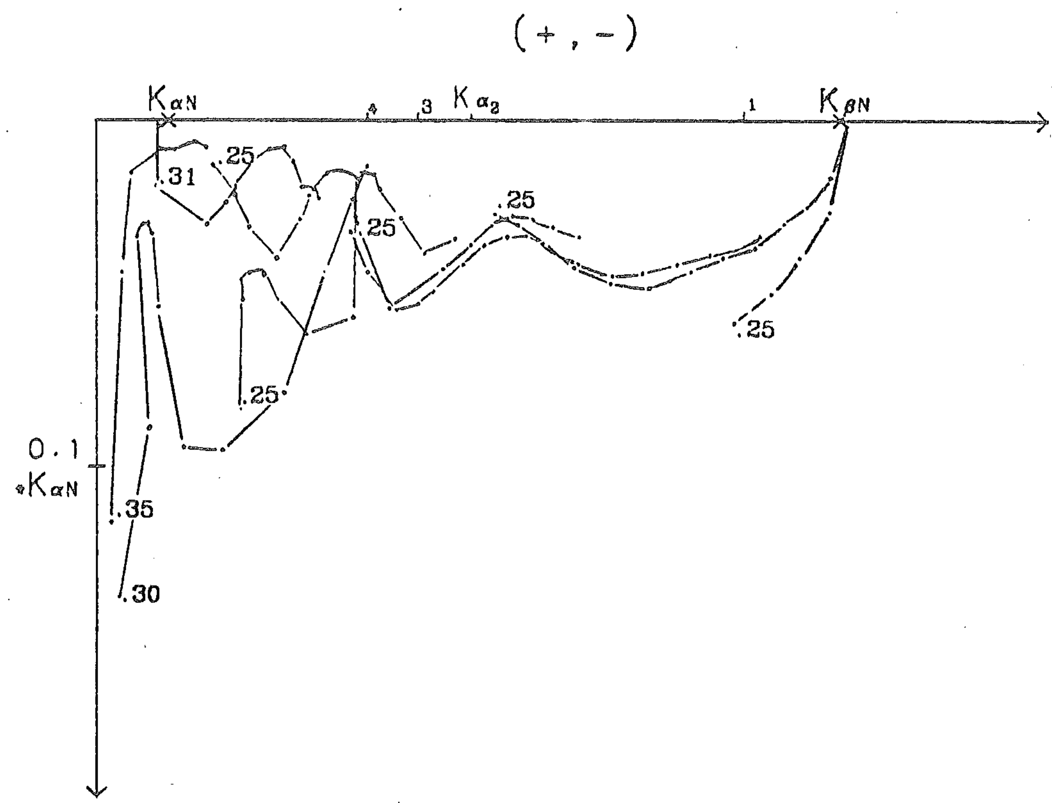


Figure 22. Same as Figure 20, but for five-layer CUS model.

the shear-coupled PL wave.

As the frequencies keep increasing, the poles will approach the shear branch point, then cross the cut, and enter the (+,+) sheet to become the normal modes. It is thus possible that, at some frequencies, the shear branch point is also a pole. One might suspect that the PL poles, when crossing the cut from (-,+) to (+,-) sheet, will give a singularity on our integration path. It is fortunate that this will not happen, because the integration is carried out on the (+,+) sheet and the cut which the PL modes cross is not the one we are integrating.

Figure 21 shows the leaky poles on the (-,-) sheet. Except around the P branch point, these poles have relatively large imaginary components. Hence we can expect that their contribution will be small. This is easy to understand since the waves from this part have large phase velocity, or equivalently travel nearly vertically. Hence the leakage of waves into the halfspace will carry away most of the energy. These poles will be named 'weak' leaky poles.

When the model becomes complex, the pole shifting also becomes complicated. Figure 22 shows the variation on the (+,-) sheet for the CUS model of Table 1. In this frequency range (0.25 to 0.4 Hz), the effect of the first weathered layer cannot be seen. The variation between k_{α_2} and k_{α_N} is rapid; however, after k_{α_2} the shear-coupled PL modes retain their simple shifting pattern.

From the above discussion, we find that there exist at least three kinds of poles which affect the values of the integrand along the real branch cut: (1) weak leaky poles between 0 and k_{α_N}, (2) PL-OP poles between k_{α_N} and k_{α_2}, (3) shear-coupled PL poles between k_{α_2} and k_{β_N}. Figures 23 and 24 strongly support this conclusion. In Figure 23 pole positions and integrand responses like those of Figure 3 are displayed, with four plots corresponding to 0.25, 0.5, 0.75 and 1.0 Hz. Different source types are arbitrarily chosen. Figure 24 shows the same response variations with different source type and the corresponding leaky poles for the CUS model at 0.25 and 0.75 Hz, respectively. As expected, the correspondence between the pole location and integrand variation is very apparent. It is obvious that the PL-OP poles are the most significant contributors to the integration, which give rise to the main part of the body waves with dominant P characteristics. The shear-coupled PL poles are important only for several components such as ZSS. These shear-coupled poles have an apparent tendency to be associated with normal poles, which can be said to possess properties which are between 'pure' surface and 'pure' body waves. The causality appearing in Figures

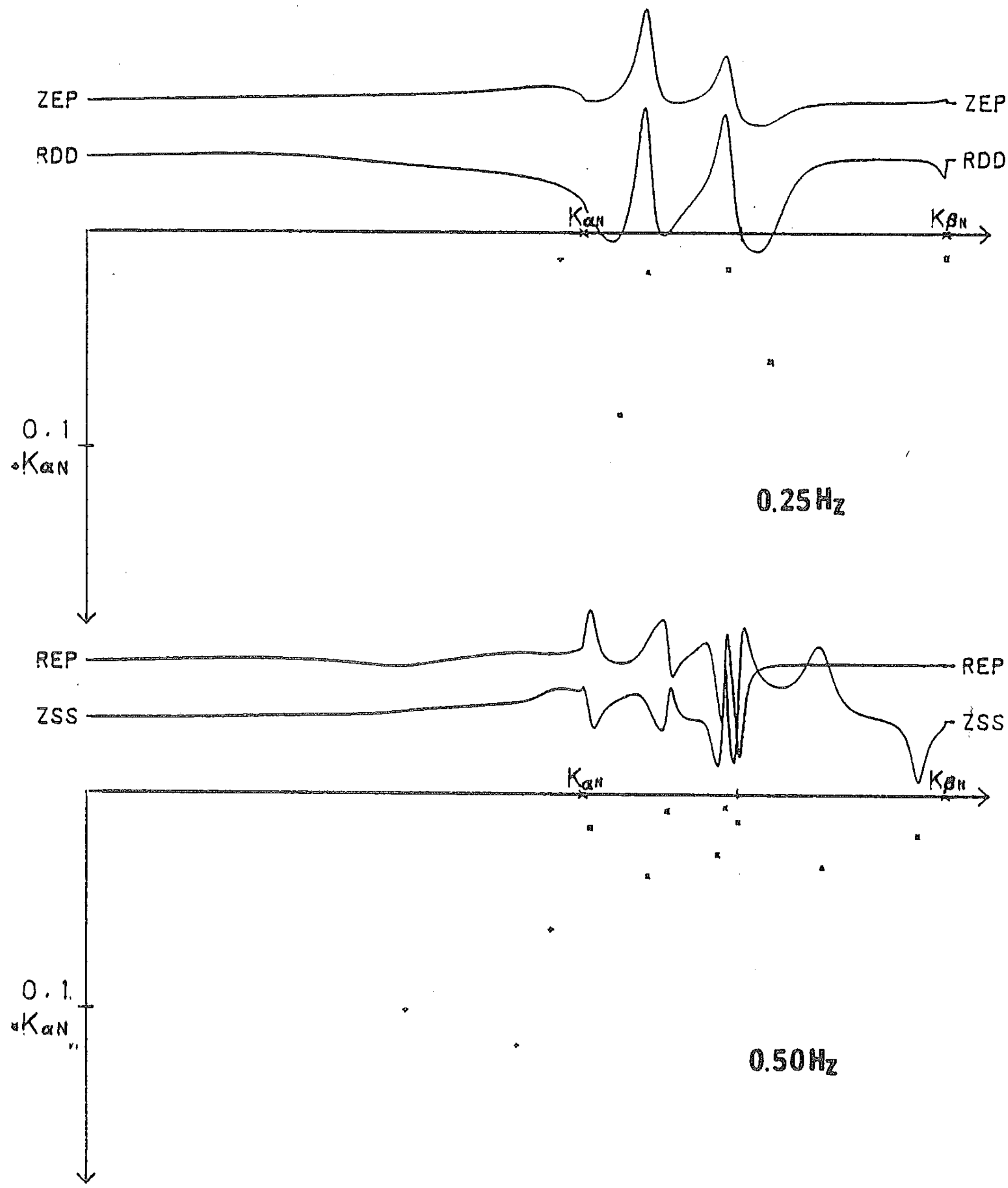
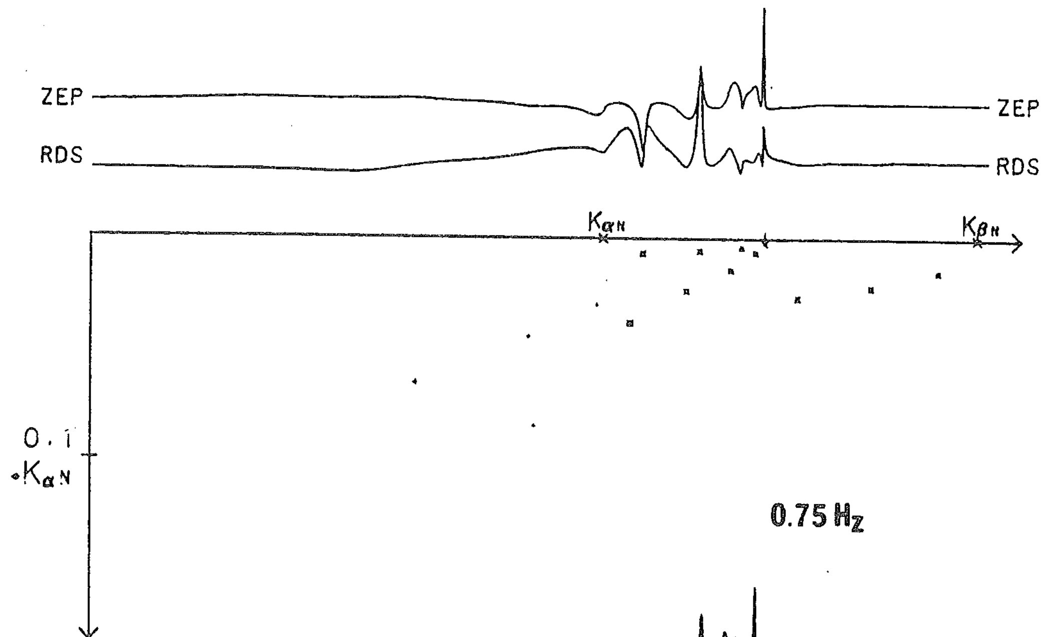


Figure 23. The effect of leaky modes on the variations of integrands along the real branch cut. 'x' denotes the modes on the (+,-) sheet and '+' denotes the modes on the (-,-) sheet. The response curves of integrands are obtained using the SCM model with the source at 10 km depth. The names of the integrand responses are the same as those in Figure 3. Four plots (two in the next page) correspond to the frequencies at 0.25, 0.50, 0.75, and 1.00 Hz, respectively.



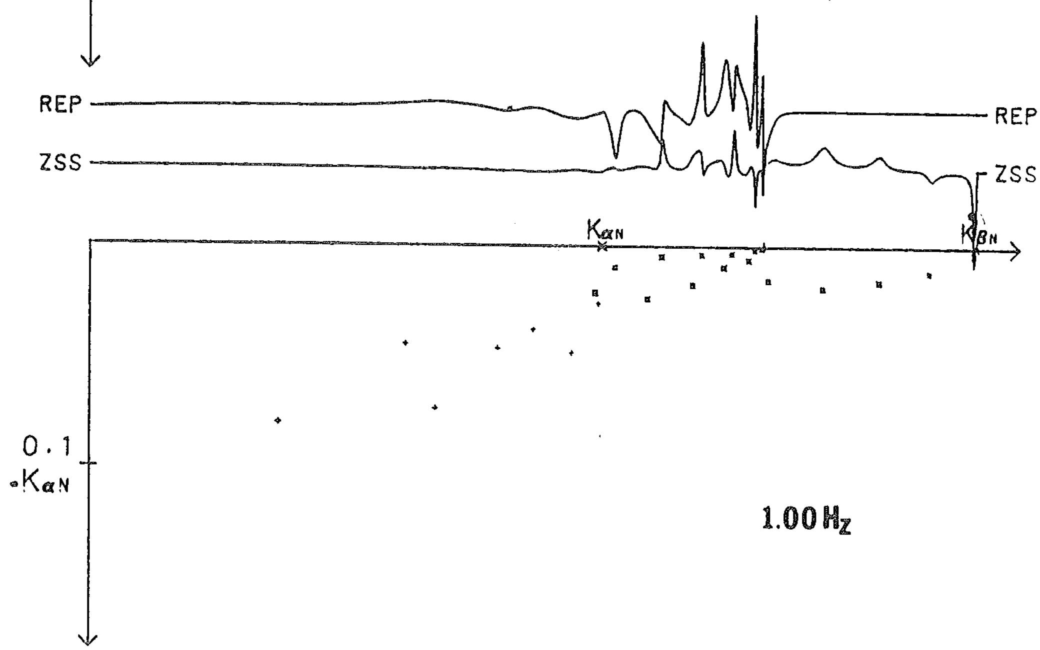


Figure 23. (cont'd)

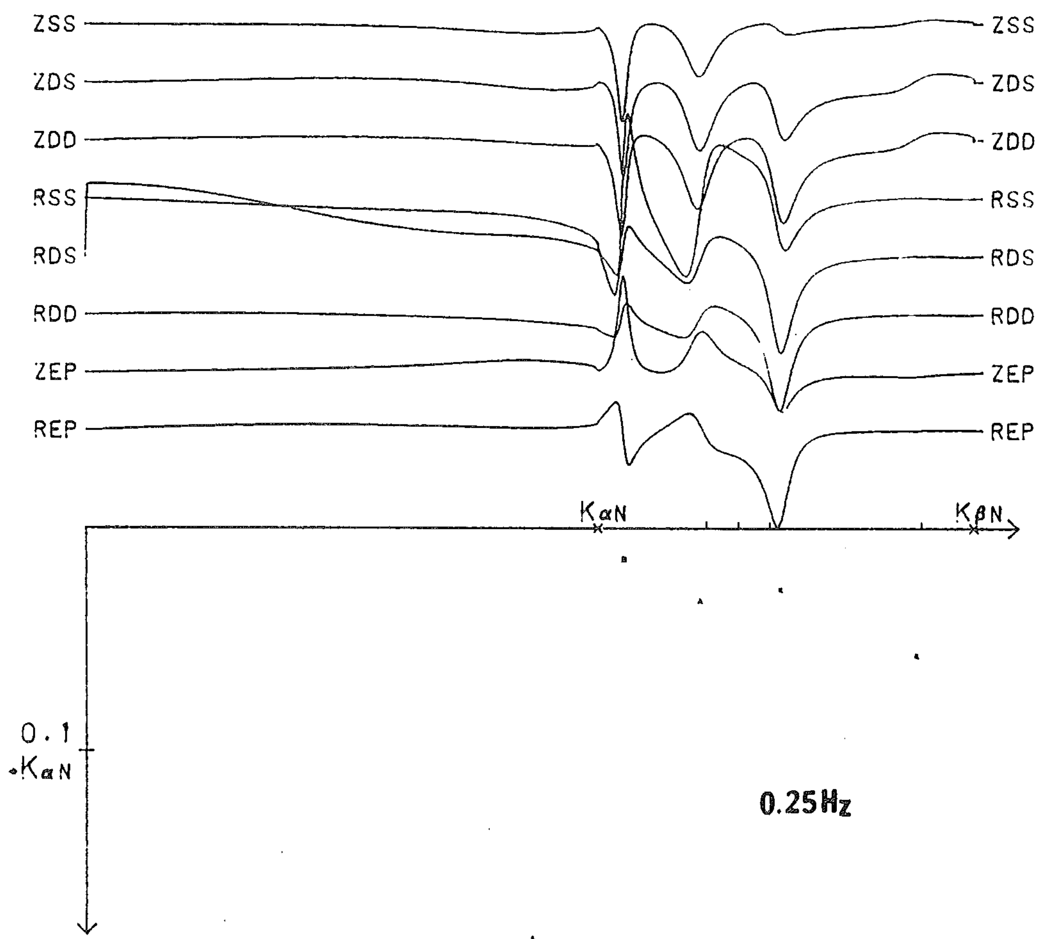


Figure 24. Results corresponding to Figure 23, but for the CUS model at the frequencies 0.25 and 0.75 Hz.

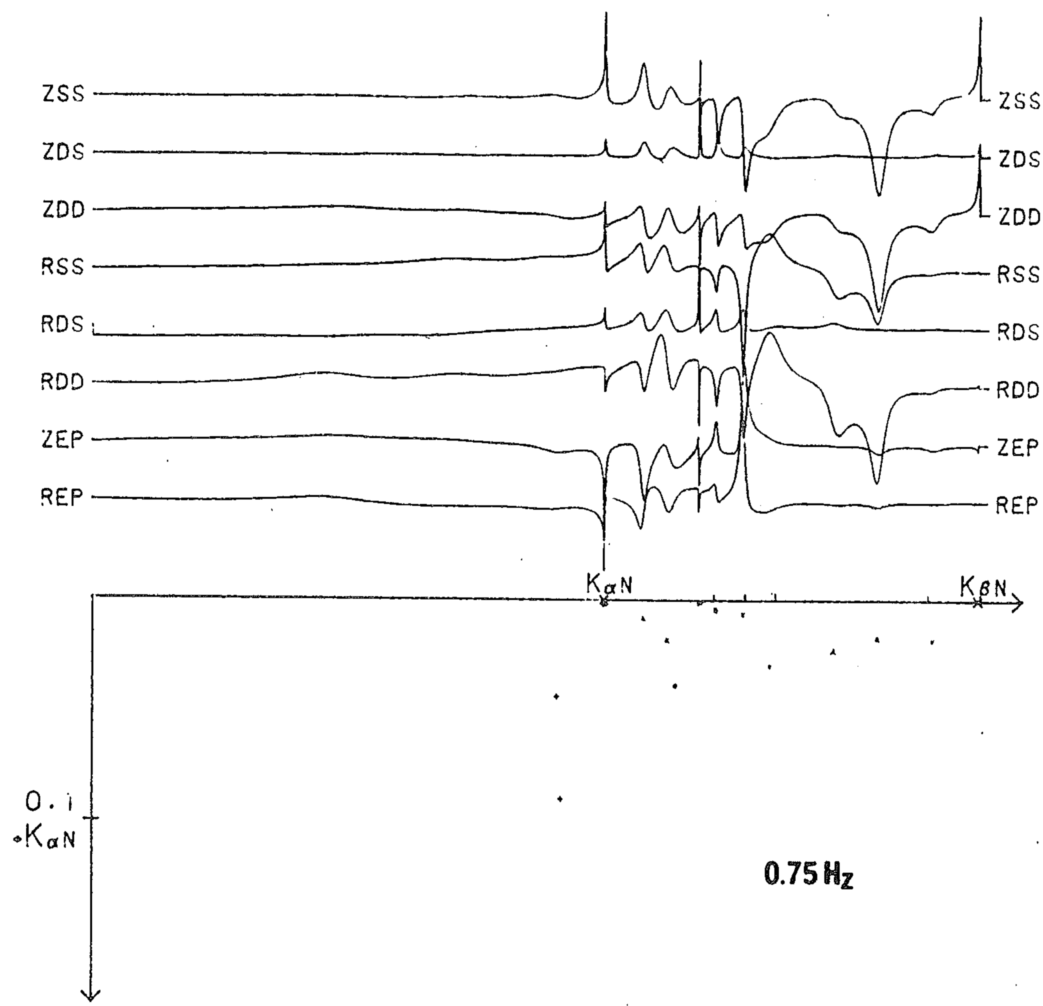


Figure 24. (cont'd)

16 to 18 arises from the contributions of these poles which compensate the Gibb's effect of abrupt truncation of normal modes at the S branch point. The contribution of weak leaky poles is relatively small, even though they might be very near the branch cut. This leads to the argument that the (-,-) cut is a stronger 'barrier' than the (+,-) cut.

For SH waves, the situation changes since there are no more PL poles. Figure 25 shows the migration of SH-OP modes, which have the properties of weak poles and shear poles in the P-SV case. Their contributions are only in compensating the noncausality from normal modes. In this sense, we might be able to say that the shear or shear-coupled leaky poles and the normal modes near the S branch point are the factors which constitute the S type body waves. In this range, it is not possible to define an absolutely distinguishing point for surface and body waves. A similar conclusion was obtained from ray expansion theory which supposes that body waves come from a finite number of rays, but when infinite rays are included they form the surface wave (Kennett, 1974). There is not a definite separation between body and surface waves.

After the study of the influence of leaky modes on the integrand, it is now clear how to perform branch line integration. The variable transform in equation

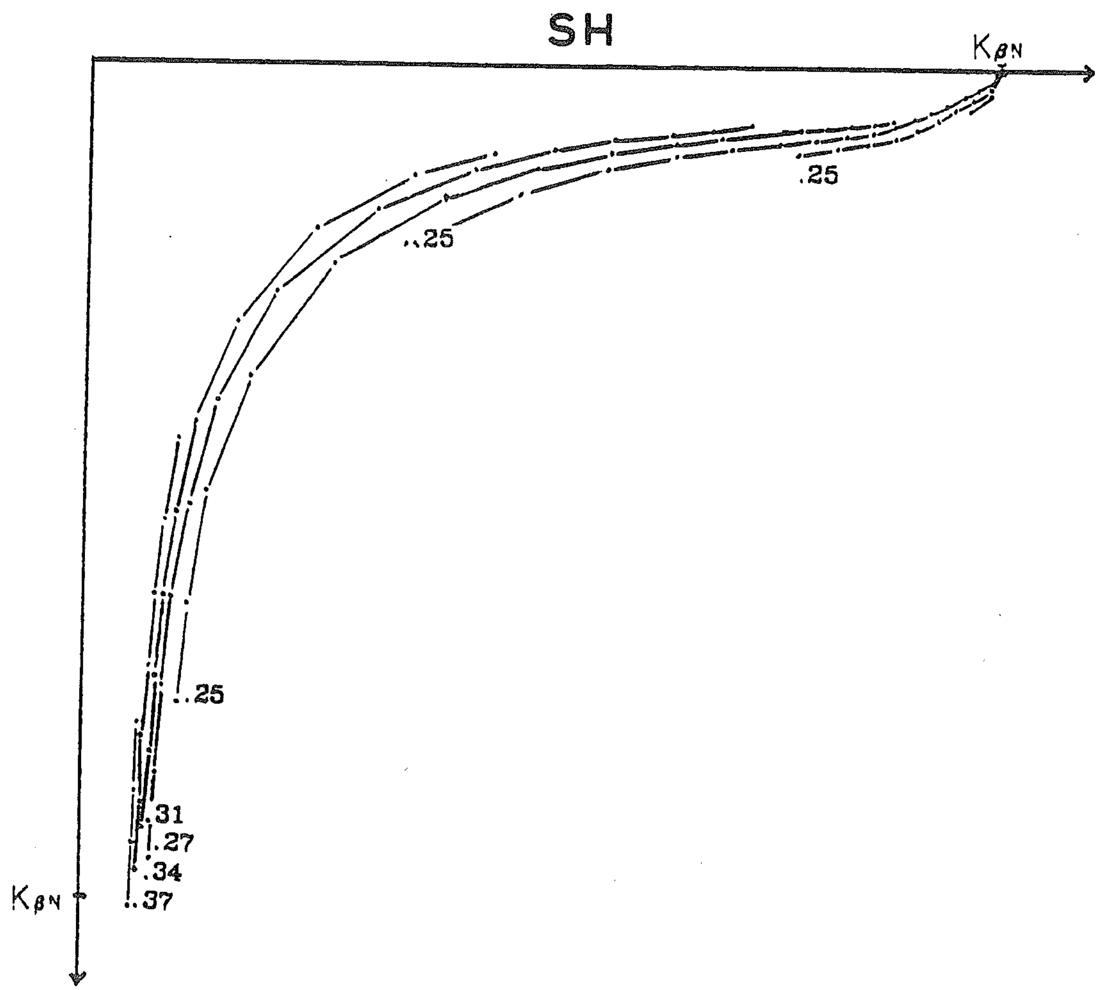


Figure 25. Paths of leaky modes of SCM model for the SH case. Other parameters are the same as Figure 20.

(II-4-5) just makes the sampling closer in the region where PL-OP modes are important, and thus improves the computational accuracy. Finally, the solution in equation (II-4-1) is determined not only from the integrand discussed above, but from the inner product of this integrand and the Hankel function. The Hankel function is an oscillatory function which depends on kr, i.e., wavenumber times epicentral distance. The final contribution to synthesizing seismograms can be looked upon as a cross-correlation between these two oscillatory functions along the real wavenumber axis. Because of the relatively smooth variation of the SH component (Figure 3), the branch line integration contribution to the tangential component seismogram is significant only at short distances. However, for P-SV waves, the effect will persist to much greater distances, as seen from Figures 16 and 17.

4.2 Locked Mode Approximation

In the previous section we have discussed the relationship between branch line integrals and leaky modes as a way to improve the evaluation of body waves. An interesting method to treat this contribution, called the locked mode approximation, was introduced by Harvey (1981). He simply added a deep rigid cap to the bottom of the structure to 'catch' leaky modes. Since the cutoff wavenumber for the locked modes is controlled by the half-space S-wave branch point, the branch cut shrinks and opens a place for more higher order 'locked' modes as the S-wave velocity in the bottom layer increases. These 'locked mode contributions' can easily be evaluated by normal mode superposition (Harkrider, 1964), which has been successfully used in synthesizing surface waves. Thus the method provides an easy-to-solve solution for the branch line integral or leaky mode contributions.

Another novel approach to the problem of branch line integration was proposed by Bouchon and Aki (1977) and Bouchon (1979, 1981). They made an attempt to discretize the wavenumber responses by presenting an infinite number of 'extra' sources at evenly spaced grid points or rings. Because of the interference of waves from these sources, the integrands are quantified and form an exact solution to be evaluated at discrete wavenumber points. This technique is beyond the scope of this dissertation. Here we just simply discuss the method of locked mode approximation which, in addition, serves as a test for our eigenfunction solutions of chapter III.

With the same methodology as used before, the integration is still made using real wavenumber and real frequency so that the search for poles is easier.

Nevertheless an anomalously high velocity cap layer is placed at the bottom of the structure. Such a process just serves to 'squeeze' the leaky modes in the lower Riemann sheets into the real 'trapped' mode position. The cap layer not only traps most of the seismic energy in the upper layers, but also brings in those pulses not existing in the real structure. If the cap layer is situated at a depth so large that the energy reaching it is small compared to the energy in the near surface layers, these false phases can be isolated and filtered by a wavenumber window (Embree et al , 1963). However for the test of normal mode theory of chapter III, we will not consider this filtering, so as to pursue the normal mode superposition method faithfully.

When using the locked mode approximation, several requirements should be fulfilled beforehand:

(1) The calculation of locked mode positions must be precise. Figure 26 shows the pole positions along the real wavenumber axis for a cap layer structure. The model is SCM listed in Table 1 with a cap layer at 240 km depth and P velocity 20 km/sec, S velocity 10 km/sec, and density 6 gm/cm^3. As the regular normal modes are still kept at the same places, the surface wave contribution is not affected. However, the newly generated 'locked' poles are numerous and very close to each other. For example, 15 modes for the original

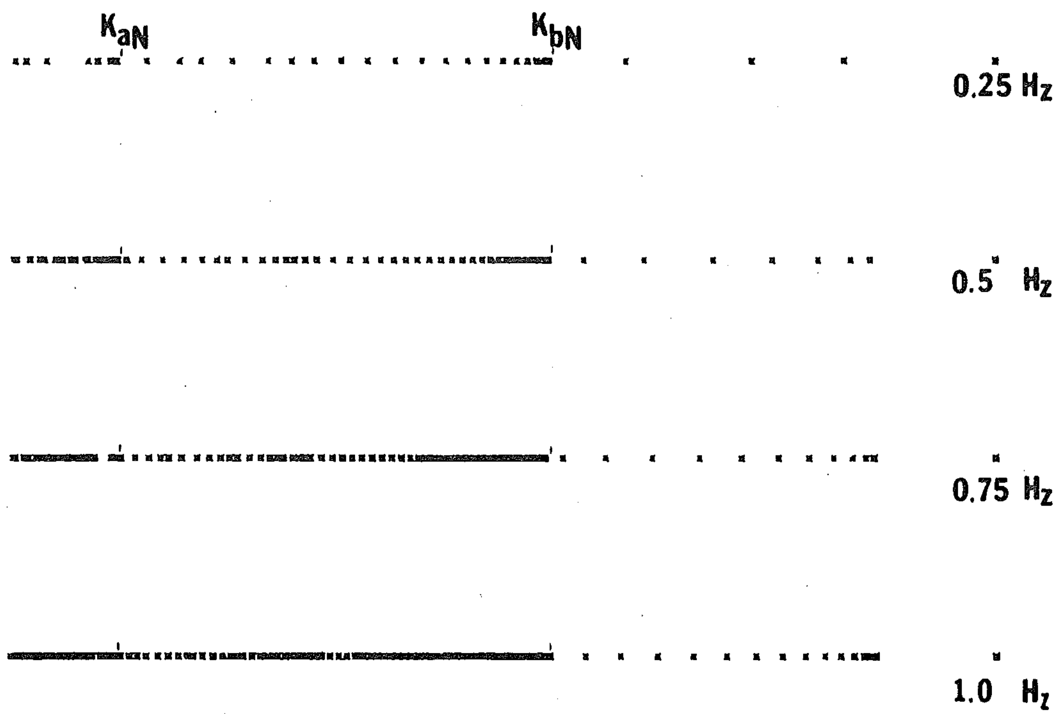


Figure 26. The positions of poles along the real k-axis at the frequencies 0.25, 0.50, 0.75, and 1.00 Hz. k_{α_N} and k_{β_N} are branch points for SCM model without the cap layer. When the cap layer is added, the leaky modes are forced to migrate into the normal mode positions as those shown to the left of $k_{\beta N}$. These created 'locked' modes are numerous and are difficult to locate. The positions of regular normal modes are essentially not affected by the present of the cap layer.

model at 1 Hz become 119 modes for the capped layer model. The treatment of these pseudo-normal modes requires special care.

(2) The pole contribution arising from the partial derivative of the period equation (or equivalently the amplitude factor defined in equation III-2-2) is not easy to evaluate since these poles have very large phase velocities and cause numerical difficulty. Thus we use equation (III-2-2) instead of the alternate expression for A_R .

(3) Because of the large number of poles involved, it is important to pay attention to the computational efficiency. In the previous section, we have found a different importance for each kind of leaky mode. Is there any difference of contribution from these locked poles? One test is shown in Figure 27, where the amplitude factors for mode order 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, and 110 are displayed. This figure reveals that each mode has about the same level of contribution. Hence, unfortunately, all of the locked modes must be taken into account.

To overcome these difficulties, a new technique other than the usual normal mode method (Harkrider, 1964; Saito, 1972) is required. Harvey (1981) used Abo-Zena's (1980) formulation to calculate displacement eigenfunctions, which in turn give the stress

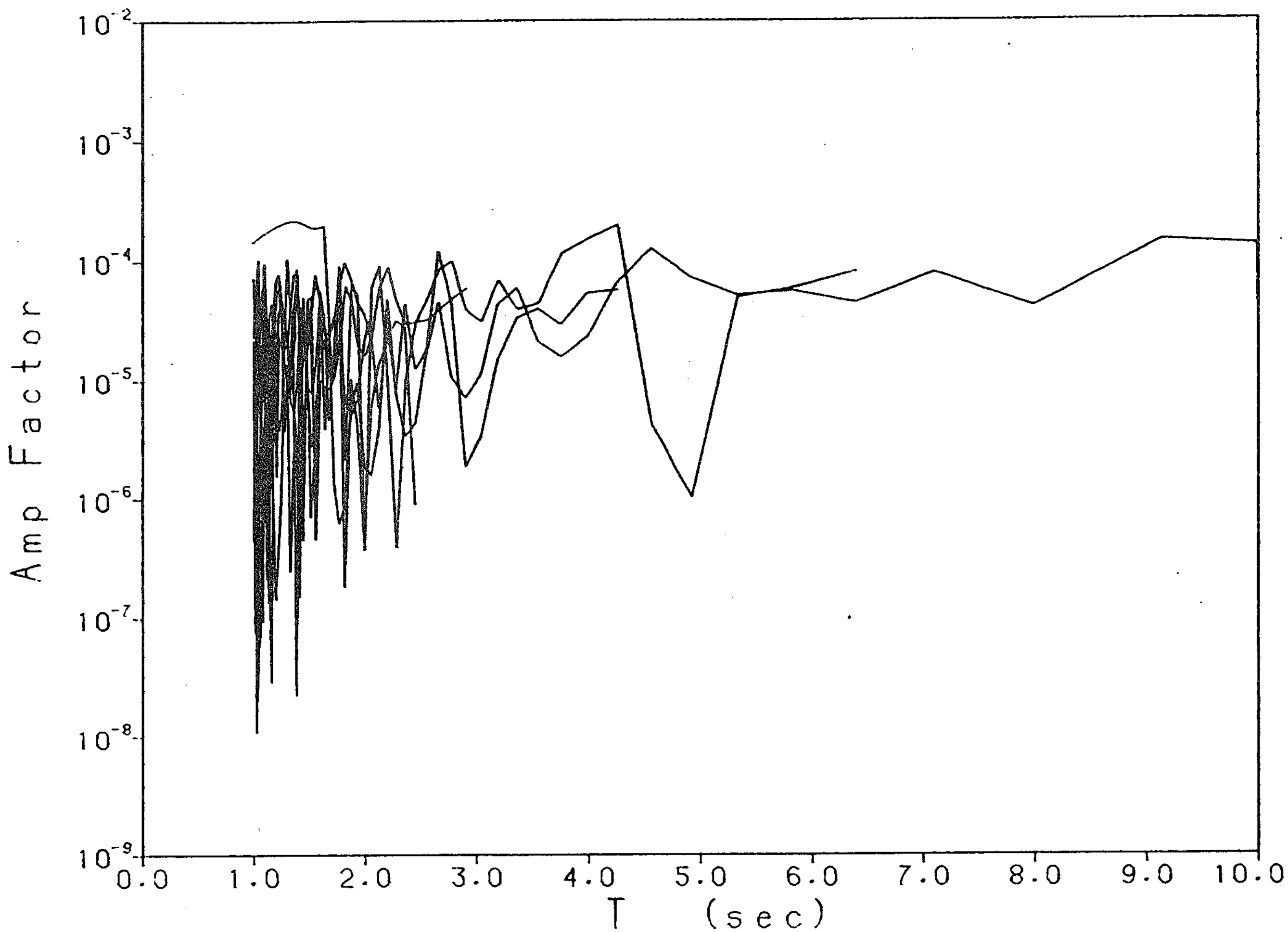


Figure 27. The amplitude factors, which represent the relative contributions of different modes to the final solution, are displayed against periods for the capped SCM model. The modes displayed are of the order 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, and 110.

eigenfunctions, by a constraint derived from the relation between the displacements and stresses. Similarly after the study of normal mode theory in chapter III, we also raised the eigenfunction theory to a very powerful stage. This formulation will be used to consider the idea of locked mode approximation. On the other hand, the locked mode approximation may provide a good test of the stability of our method. Figures 28 to 30 present the results.

Figure 28 illustrates the results from methods of chapters II and III, as compared to the locked mode approximation. In this figure, (a) is the locked mode approximation, (b) the branch line integration and pole, (c) the normal pole contribution, and (d) the branch line integral. The display is the ZSS component due to a point source at 10 km depth in the SCM model with highest frequency 1 Hz and source time function (equation II-4-8) τ = 0.5 sec. The agreement between (a) and (b) is excellent except for some artificial early arrivals due to the fact that the cap layer effectively introduces a sharp wavenumber cutoff in the solution. This means that the methods we developed in chapters II and III can be trusted. Figure 29 is the RDS component for the same case, and Figure 30 gives the TDD component. The match of results from the locked mode approximation to the complete seismogram is obvious. However the computation times are quite

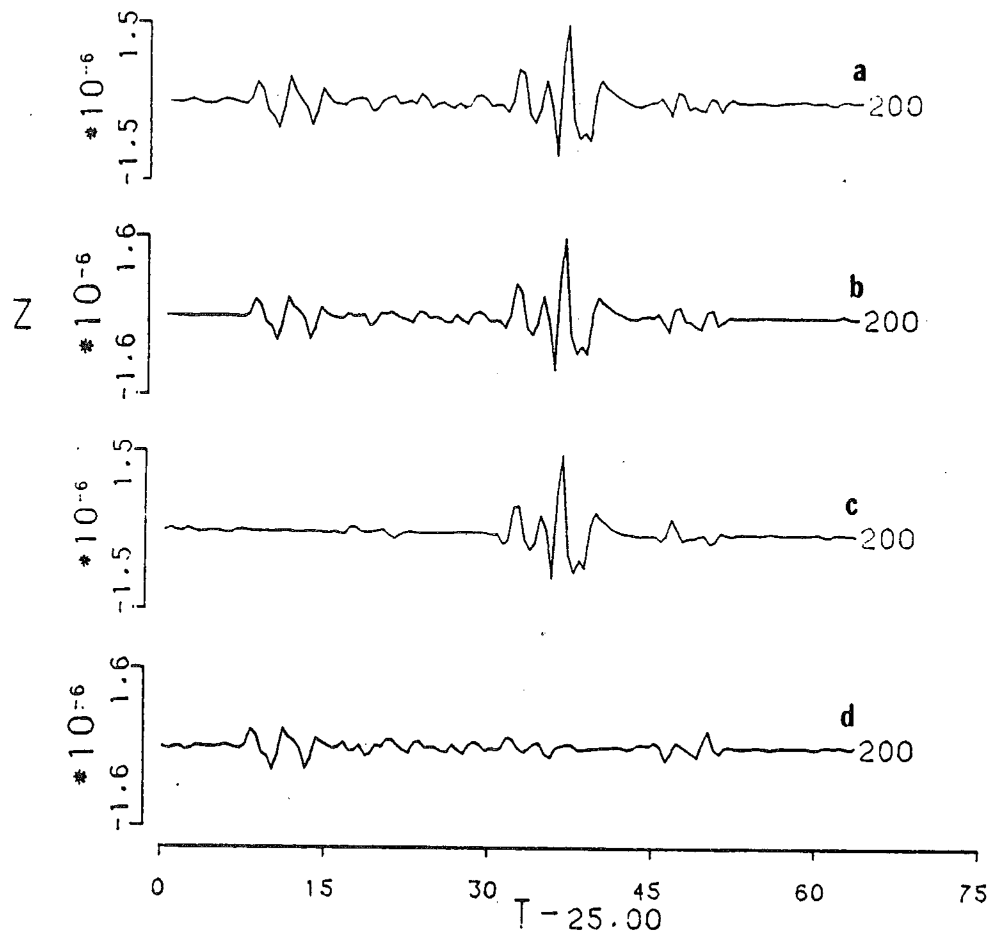


Figure 28. Comparison of locked mode approximation to the complete solution using the method of chapter 2. (a) the locked mode approximation solution; (b) the wave integral complete solution; (c) the pole contribution; (d) the branch line integral contribution. The SCM model with a strike-slip dislocation source at the depth of 10 km is used. The cap layer is located at 200 km deep and has a P velocity of 20 km/sec, S velocity of 10 km/sec, and density of 6 gm/cm^3.

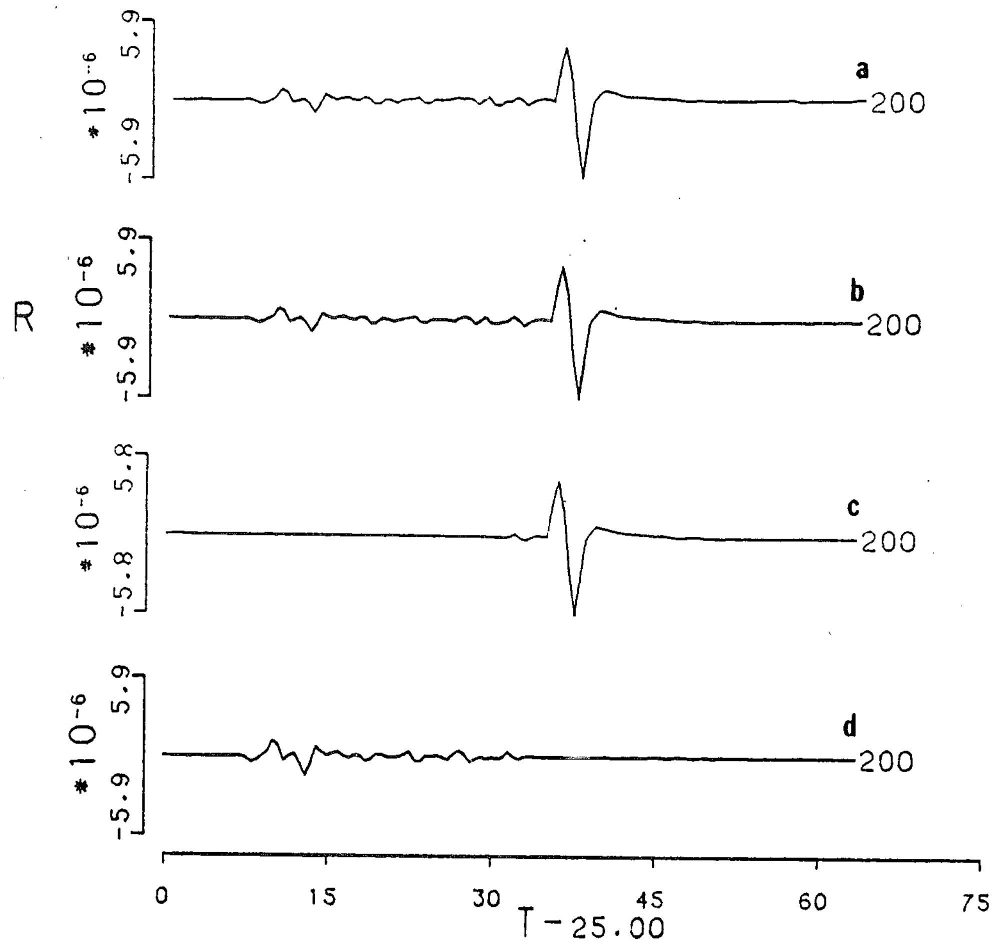


Figure 29. Results corresponding to Figure 28, but for the radial component and a dip-slip dislocation source.

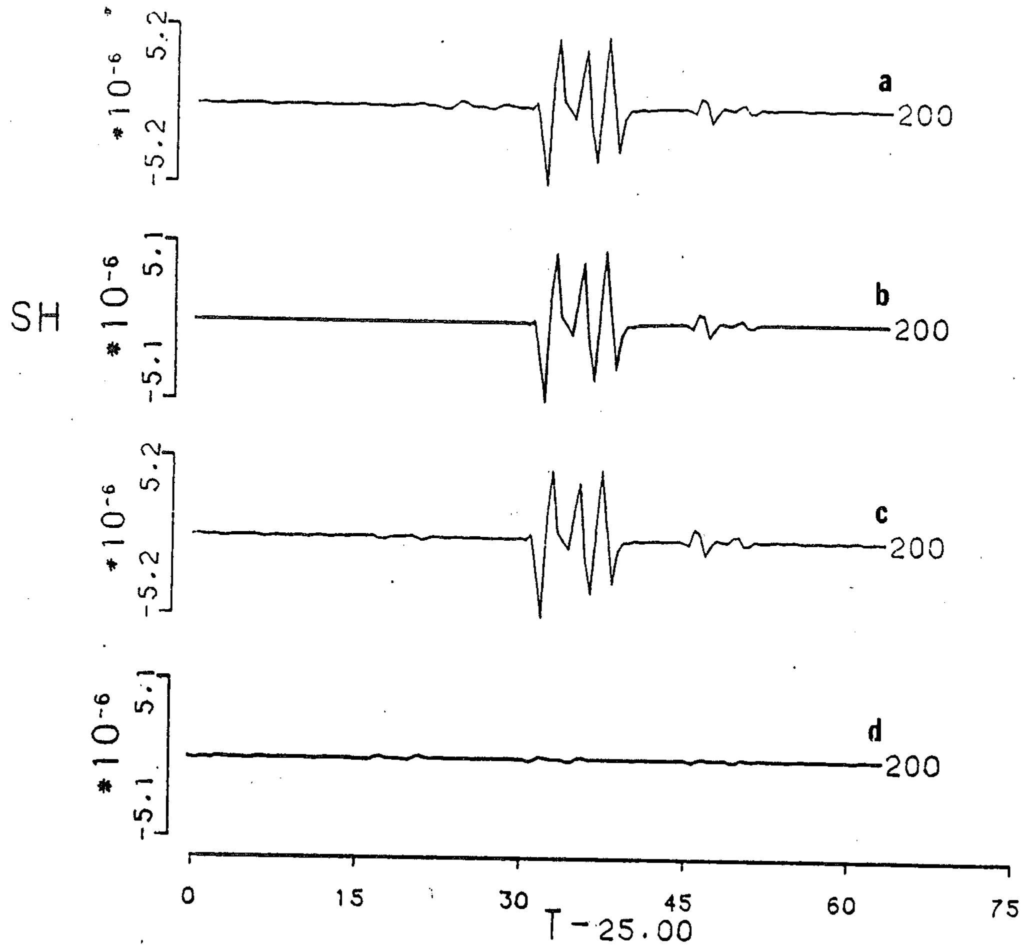


Figure 30. Results corresponding to Figure 28, but for the tangential component and a 45° dip-slip source.

different. Using the locked mode approximation the calculation takes about 2 hours on a DEC PDP 11/70 minicomputer, while using the direct integration method requires only half an hour. Since these two methods use the same wave theory, any time consuming aspect, such as the case of high frequencies, will be required for both methods. The advantage of the locked mode approach is that once the dispersion curves are found for a given model, they do not have to be recomputed.

4.3 Reflection Method and Reflectivity Method

In the previous chapters, we built up the whole wave field by summing the layer responses over wavenumber and frequency. The calculation includes all details of energy possibly excited in the layered medium. Sometimes, for the study of particular portions of the structure, we desire to suppress the reflections from certain interfaces. This is especially useful for body wave studies. This section will develop a method to 'eliminate' the reflections from some layer boundaries. The significance of the pole contribution to the generation of surface waves becomes apparent if the reflections from the free surface, the strongest reflector, are artificially suppressed. Two theories closely related to this section are worth reviewing. One is Kennett's reflection and transmission coefficient method, which is referred to as the 'reflection method' here (Kennett and Kerry, 1979), and the other is Fuchs's reflectivity method (Fuchs and Müller, 1971). In this section, these methods will be stated using the terminology of this dissertation.

Reflection Method

The reflection method introduced by Dr. Kennett and his colleagues (Kennett, 1974; Kennett et al, 1978; Kennett and Kerry, 1979; Kennett, 1980; and Kerry, 1981) possesses properties of both ray theory and wave theory. They established a connection between conventional matrix methods and the reflection and transmission properties of a single interface. This approach lends itself to a ray interpretation, however, from a view point of gross reflection and transmission response of layers. Starting from the response of the entire stack of layers, the Haskell matrix R in equation (II-2-6):

$$R = E_N^{-1} \alpha_{N-1} \cdots \cdots \alpha_1 .$$

α's, the layer matrices, can be expressed in terms of the fundamental matrix E and phase matrix Λ (equation II-1-18),

$$\alpha_n = E_n \Lambda_n E_n^{-1} .$$

Hence,

$$R = (E_N^{-1} E_{N-1}) \Lambda_{N-1} (E_{N-1}^{-1} E_{N-2}) \cdots \cdots$$
$$\cdots \Lambda_n (E_n^{-1} E_{n-1}) \Lambda_{n-1} \cdots (E_2^{-1} E_1) \Lambda_1 E_1^{-1} \quad . \qquad \text{(IV-3-1)}$$

The reflection and transmission can only occur at velocity discontinuities; therefore only the terms within the parenthesis of equation (IV-3-1) contain the information of wave-interface interaction. The matrix Λ is only used to 'phase relate' the waves through the layer. Consider a single interface. The potential-constant vector K , which represents the waves inside the layer, can be rewritten as

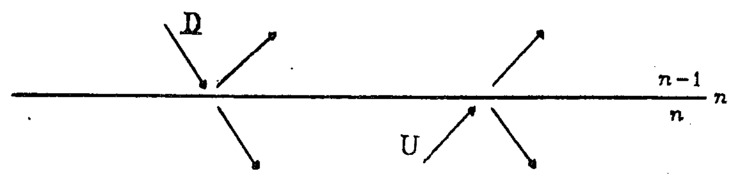
$$K_n = \begin{bmatrix} A' \\ B' \\ A'' \\ B'' \end{bmatrix}_n = \begin{bmatrix} U \\ \cdots \\ D \end{bmatrix}_n ,$$

where U and D are 2x1 matrices for the P-SV case containing upgoing P,SV and downgoing P,SV potentials, respectively. For the SH case, U and D are scalars. The waves just above and below the interface satisfy

$$\begin{bmatrix} U \\ \cdots \\ D \end{bmatrix}_n = E_n^{-1} E_{n-1} \begin{bmatrix} \underline{U} \\ \cdots \\ \underline{D} \end{bmatrix}_{n-1} = F_n \begin{bmatrix} \underline{U} \\ \cdots \\ \underline{D} \end{bmatrix}_{n-1} .$$

The underscoring denotes the quantities at the bottom of the n-1 layer. F_n is called the reflection-and-transmission matrix. It is noted that F_n does not contain the phase terms. These terms, as discussed in section 2.2, might be exponentially growing and cause an accuracy problem.

Consider a downgoing wave incident at the boundary,



we define

$$\mathbf{U}_{n-1} = \mathbf{r}_D \, \mathbf{D}_{n-1}$$
$$\mathbf{D}_n = \mathbf{t}_D \, \mathbf{D}_{n-1} \quad .$$

$\mathbf{r}_D$ and $\mathbf{t}_D$ are matrices of the reflection and transmission coefficients for both P and SV waves in downward propagation,

$$\mathbf{r}_D = \begin{bmatrix} r_{pp}^D & r_{sp}^D \\ r_{ps}^D & r_{ss}^D \end{bmatrix} \qquad \mathbf{t}_D = \begin{bmatrix} t_{pp}^D & t_{sp}^D \\ t_{ps}^D & t_{ss}^D \end{bmatrix} \quad .$$

Similarly, an upgoing wave incident at the boundary defines $\mathbf{r}_U$, $\mathbf{t}_U$. Following the discussion of Kennett (1974), or more clearly Frasier (1970), we have

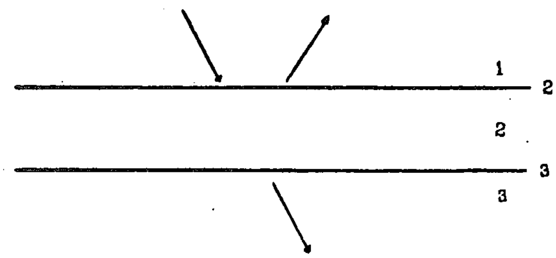
$$\mathbf{F}_n = \mathbf{E}_n^{-1}\mathbf{E}_{n-1} = \left[\begin{array}{c:c} \mathbf{t}_U^{-1} & -\mathbf{t}_U^{-1}\mathbf{r}_D \\ \hdashline \mathbf{r}_U \mathbf{t}_U^{-1} & \mathbf{t}_D - \mathbf{r}_U \mathbf{t}_U^{-1} \mathbf{r}_D \end{array} \right]_n \qquad \text{(IV-3-2)}$$

and

$$\begin{aligned} \mathbf{r}_D &= -\,\mathbf{F}_{11}^{-1}\mathbf{F}_{12} \\ \mathbf{t}_D &= \mathbf{F}_{22} - \mathbf{F}_{21}\mathbf{F}_{11}^{-1}\mathbf{F}_{12} \\ \mathbf{r}_U &= \mathbf{F}_{21}\mathbf{F}_{11}^{-1} \\ \mathbf{t}_U &= \mathbf{F}_{11}^{-1} \quad . \end{aligned} \qquad \text{(IV-3-3)}$$

Note that $det(F_{11})$ is the Stoneley function.

If the waves pass through a layer from the bottom of layer 1 to the top of layer 3;



the response for potential-constant vectors through the layer becomes

$$(E_3^{-1} E_2) \Lambda_2 (E_2^{-1} E_1) \ .$$

The situation does not become complex, since an iterative form can be found (Kennett, 1974):

$$\begin{aligned}
r_{D_{3\underline{1}}} &= r_{D_2} + t_{U_2}(P^{-1} r_{D_3} P^{-1}) \left[1 - r_{U_2} (P^{-1} r_{D_3} P^{-1}) \right]^{-1} t_{D_2} \\
t_{D_{3\underline{1}}} &= (t_{D_3} P^{-1}) \left[1 - r_{U_2} (P^{-1} r_{D_3} P^{-1}) \right]^{-1} t_{D_2} \\
r_{U_{3\underline{1}}} &= r_{U_3} + (t_{D_3} P^{-1}) r_{U_2} \left[1 - (P^{-1} r_{D_3} P^{-1}) r_{U_2} \right]^{-1} (P^{-1} t_{U_3}) \\
t_{U_{3\underline{1}}} &= t_{U_2} \left[1 - (P^{-1} r_{D_3} P^{-1}) r_{U_2} \right]^{-1} (P^{-1} t_{U_3})
\end{aligned} \qquad \text{(IV-3-4)}$$

where

$$P = \begin{bmatrix} e^{\nu_\alpha d_2} & 0 \\ 0 & e^{\nu_\beta d_2} \end{bmatrix} ,$$

and $3\underline{1}$ represents the stack between the top of layer 3 and the bottom of layer 1. In equation (IV-3-4) we have used P to phase relate the reflection and transmission coefficients at interface 3 to interface

2. However, note that r_{U_3} need not be 'phase related'. Equation (IV-3-4) reveals some interesting features:

(1) The term $[1-r_U(P^{-1}r_D P^{-1})]^{-1}$ can be expanded as a power series;

$$[1-r_U(P^{-1}r_D P^{-1})]^{-1} = 1 + r_U(P^{-1}r_D P^{-1}) + r_U(P^{-1}r_D P^{-1})\, r_U\, (P^{-1}r_D P^{-1}) + \cdots$$

which describes the ray propagation with different internal reflections and transmissions in the layer. Kennett (1980) has provided a detailed explanation about this ray viewpoint of wave theory. Hence, the total response of layers is just a superposition of multiple reflections from boundaries.

(2) The phase term which might grow exponentially is excluded from the calculation of reflection and transmission coefficient at layer boundaries. Hence the problem of loss of precision can be well controlled.

(3) To calculate R , we need only start at the base of the layers, progress upward, iteratively adding a layer to the stack, one at a time. This iterative approach represents the most important step of Kennett's method.

Suppose that the stacking from the half-space to the top of first layer (just beneath the surface) has been accomplished. The (1,1) component of 2x2 partition of matrix R can be determined from

$$\mathbf{R} = \begin{bmatrix} \mathbf{t}_{\bar{U}}^{-1} & -\mathbf{t}_{\bar{U}}^{-1}\mathbf{r}_D \\ \cdot & \cdot \end{bmatrix}_{N1} \begin{bmatrix} \overline{\mathbf{E}}_{11} \\ \overline{\mathbf{E}}_{21} \end{bmatrix}_1 ,$$

i.e.,

$$\mathbf{R}_{11} = \mathbf{t}_{\bar{U}_{N1}}^{-1} (\overline{\mathbf{E}}_{11} - \mathbf{r}_D \overline{\mathbf{E}}_{21}) ,$$

where $\overline{\mathrm{E}}$ represents the inverse of the E matrix of the first layer. Going back to our formulas for the displacement at the free surface (equation II-2-5),

$$\begin{bmatrix} 0 \\ \mathbf{D} \end{bmatrix}_N = \mathbf{X}\,\mathbf{S} + \begin{bmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} \\ \mathbf{R}_{21} & \mathbf{R}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{W}_1 \\ 0 \end{bmatrix}_1$$

or

$$\mathbf{W}_1 \equiv \begin{bmatrix} U_{r_1} \\ U_{z_1} \end{bmatrix} = - \mathbf{R}_{11}^{-1} \mathbf{X}_{1i} \mathbf{S}_i \quad . \qquad \text{(IV-3-5)}$$

If we use the discontinuity in the constant - potential vector to describe the source (their explicit form will be listed in section 5.1) rather than a discontinuity in displacement and stress,

$$\begin{aligned} \mathbf{X}\,\mathbf{S} &= \mathbf{E}_{\bar{N}}^{-1} \mathbf{E}_{N-1} \Lambda_{N-1} \mathbf{E}_{\bar{N}-1}^{-1} \cdots \Lambda_m (\mathbf{E}_m^{-1} \mathbf{S}) \\ &= (\mathbf{E}_{\bar{N}}^{-1} \mathbf{E}_{N-1} \Lambda_{N-1} \cdots \Lambda_m) \Sigma_m \\ &= \begin{bmatrix} \mathbf{t}_{\bar{U}}^{-1} & -\mathbf{t}_{\bar{U}}^{-1}\mathbf{r}_D \\ & \end{bmatrix}_{NS} \begin{bmatrix} \Sigma_U \\ \Sigma_D \end{bmatrix}_m \end{aligned}$$

and equation (IV-3-5) becomes

$$\mathbf{W}_1 = -\left(\overline{\mathbf{E}}_{11} - \mathbf{r}_{D_{N1}}\overline{\mathbf{E}}_{21}\right)^{-1} \mathbf{t}_{U_{N1}}\, \mathbf{t}_{U_{NS}}^{-1} \left(\Sigma_U - \mathbf{r}_{D_{NS}}\, \Sigma_D\right). \qquad \text{(IV-3-6)}$$

$det\,(\overline{\mathbf{E}}_{11} - \mathbf{r}_{D_{N1}}\overline{\mathbf{E}}_{21})$ is the period equation. Note that, except for the source depth, any partition of layer stacking at other places such as receiver depth has been avoided. There exist several different forms of the period equation in the derivation (Kennett, 1974; Kennett, 1980; Kennett and Kerry, 1979; and Kerry, 1981). After lengthly discussion, Kerry (1981) finally chose this form to calculate dispersion values. However, by using a different partition of layer stacking, he was able to find an interesting screen effect of a low velocity zone at depth.

We will not expand equation (IV-3-6) to calculate seismograms. Only the concept of reflection and transmission coefficients or equation (IV-3-2) will be used in the following development for the reflection suppression technique.

Reflectivity Method

The reflectivity method is an important application of the Haskell matrix (Fuchs, 1968; Fuchs and Müller, 1971; Kind and Müller, 1975; Müller and Kind, 1976; and Kind, 1976, 1978, 1979), hitherto mostly used for body wave computation for an explosive type of source. This method involves a double integration with respect to real wavenumber and real frequency, but over a small range of interest such as those with real number of incident angle. Such a numerical integration corresponds only to our real branch line integral. The integration could cover the whole range of response including the surface-wave poles by adding a small complex part to the velocity in each layer, accounting for anelasticity, which would move poles off the positive k-axis (Kind, 1979). However, one has to sample the integrands very closely over a much larger range of angle of incidence, and the computation time increases substantially. The reason for dense sampling is similar to that which we have discussed in section 4.1, i.e., the influence of leaky or normal modes on the integration along the real wavenumber axis. In principle, the reflectivity method is not suitable for the computation of normal modes.

Consider the potential-constant vector K just beneath the free surface. If there is no reflection at the free surface, we can simply make $D_1 = 0$, and the free surface displacement is just $W_1 = E_{11} U_1$, where E_{11} is, for P-SV, the (1,1) component of the 2 by 2 partition of the Haskell E matrix of the top layer, and for SH, a scalar. Fuchs (1968) used this form to treat the free surface in the original theory of the reflectivity method. However, if we take into account the free surface effect, but do not allow the wave to be reflected back into the earth, then from equation (II-1-15),

$$\begin{bmatrix} \mathbf{W}_1 \\ \cdots \\ 0 \end{bmatrix} = \begin{bmatrix} \mathbf{E}_{11} & \vdots & \mathbf{E}_{12} \\ \cdots & & \cdots \\ \mathbf{E}_{21} & \vdots & \mathbf{E}_{22} \end{bmatrix}_1 \begin{bmatrix} \mathbf{U} \\ \cdots \\ \mathbf{D} \end{bmatrix}_1 ,$$

and the solution is

$$\mathbf{W}_1 = (\mathbf{E}_{11} - \mathbf{E}_{12}\, \mathbf{E}_{22}^{-1}\, \mathbf{E}_{21})\, \mathbf{U}_1 = \mathbf{F}_1\, \mathbf{U}_1 \quad . \qquad \text{(IV-3-7)}$$

This is the form Fuchs and Müller (1971) used. For SH such a modification just doubles the amplitude, and for P-SV some conversions between P and SV take place but without much effect.

For a reflecting and transmitting interface, we can simply set $\mathbf{r}_U$ and $\mathbf{r}_D$ equal to zero to suppress its 'reflectivity'. From equation (IV-3-2) the reflection-and-transmission matrix F becomes

$$\mathbf{F}_n = \mathbf{E}_n^{-1}\, \mathbf{E}_{n-1} = \begin{bmatrix} \mathbf{t}_U^{-1} & \vdots & 0 \\ \cdots & & \cdots \\ 0 & \vdots & \mathbf{t}_D \end{bmatrix}_n .$$

Furthermore under this condition, (IV-3-3) provides that

$$\mathbf{t}_U^{-1} = \mathbf{F}_{11} \ .$$

Because of the reciprocity relation, Kennett <u>et al</u> (1978) and Fraiser (1970) found that

$$\mathbf{t}_D = \mathbf{t}_U^T \times (\, normalization \ \ factors \,) \ .$$

In our system, the normalization factors, as discussed in deriving equation (II-1-12), enter the result as

$$\mathbf{t}_D = \begin{bmatrix} t_{pp}^D & t_{sp}^D \\ t_{ps}^D & t_{ss}^D \end{bmatrix} = \begin{bmatrix} t_{pp}^U a & t_{ps}^U b \\ t_{sp}^U c & t_{ss}^U d \end{bmatrix}$$

with

$$a = \frac{\nu_\alpha'/\rho'}{\nu_\alpha/\rho} \quad b = \frac{\nu_\beta' k^2/\rho'}{\nu_\alpha/\rho} \quad c = \frac{\nu_\alpha'/\rho'}{\nu_\beta k^2/\rho} \quad d = \frac{\nu_\beta'/\rho'}{\nu_\beta/\rho} ,$$

where primes denote the quantities just above the interface and the unprimed quantities are just beneath the interface. The matrix F takes the form:

$$\mathbf{F} = \begin{bmatrix} F_{11} & F_{12} & 0 & 0 \\ F_{21} & F_{22} & 0 & 0 \\ 0 & 0 & \frac{F_{22}}{\Delta}a & -\frac{F_{21}}{\Delta}b \\ 0 & 0 & -\frac{F_{12}}{\Delta}c & \frac{F_{11}}{\Delta}d \end{bmatrix} \tag{IV-3-8}$$

where

$$\Delta = \mathbf{F}|_{12}^{12} .$$

Now let us stack a whole set of layers together. Use equation (II-2-5)

$$\begin{aligned} \mathbf{K}_N &= \mathbf{X}\,\mathbf{S} + \mathbf{R}\,\mathbf{B}_1 \\ &= \mathbf{X}\,\mathbf{S} + (\mathbf{R}\,\mathbf{E}_1)\,(\mathbf{E}_1^{-1}\mathbf{B}_1) = \mathbf{X}\,\mathbf{S} + \mathbf{Q}\,\mathbf{K}_1 , \end{aligned}$$

where

$$\begin{aligned} \mathbf{Q} &= \mathbf{R}\,\mathbf{E}_1 = \mathbf{X}\,\mathbf{Z}\,\mathbf{E}_1 = \mathbf{X}\,\mathbf{P} \\ &= \mathbf{E}_N^{-1}\,\mathbf{E}_{N-1}\,\Lambda_{N-1}\,\mathbf{E}_{N-1}^{-1}\cdots\,\Lambda_1 \\ \mathbf{P} &= \mathbf{E}_m\,\Lambda_m\,\mathbf{E}_m^{-1}\cdots\,\Lambda_1 . \end{aligned}$$

Following the same procedures between equation (II-2-6) and (II-2-8); we have

$$\mathbf{U}_1 = \begin{bmatrix} -S_i X|_{ij}^{12} P_{j2} \\ \\ S_i X|_{ij}^{12} P_{j1} \end{bmatrix} \Big/ Q\,|_{12}^{12}$$

and the free surface displacements are

$$\begin{bmatrix} U_{r_1} \\ U_{z_1} \end{bmatrix} = \mathbf{F}_1 \begin{bmatrix} -S_i X \,|_{ij}^{12} P_{j2} \\ S_i X \,|_{ij}^{12} P_{j1} \end{bmatrix} \Big/ Q \,|_{12}^{12} \qquad \text{(IV-3-9)}$$

where

$$\mathbf{F}_1 = \mathbf{E}_{11} - \mathbf{E}_{12}\,\mathbf{E}_{22}^{-1}\,\mathbf{E}_{21} = \begin{bmatrix} -\dfrac{2\gamma\nu_\alpha\nu_\beta}{k} & -2k\,\nu_\beta(\gamma-1) \\ 2\dfrac{\nu_\alpha}{\rho}(\gamma-1) & \dfrac{2\gamma\nu_\alpha\nu_\beta}{\rho} \end{bmatrix}_1 \Big/ \rho_1\left(\frac{\gamma\nu_\alpha\nu_\beta}{k^2} - (\gamma-1)^2\right)_1 .$$

To calculate Q , the form (IV-3-8) is used if the reflection is to be suppressed. If it is not to be suppressed, the layer matrix $\boldsymbol{a}$ can be used. In the transmission zone of the reflectivity method, all of the reflections are suppressed, and in the reflection zone, the normal case is retained. Hence, Q in the reflectivity method is

$$\mathbf{Q} = \mathbf{E}_N^{-1}\,\boldsymbol{a}_{N-1}\cdots\,\boldsymbol{a}_n\,\Lambda_{n-1}\,\mathbf{F}_{n-1}\cdots\Lambda_2\,\mathbf{F}_2\,\Lambda_1 \; .$$

Some aspects of our system for the reflectivity method are interesting to point out:

(1) It is just a simplified case for our system but is a generalization of the reflectivity method.

(2) The conversion of waves during passage through the interface is involved. If such a conversion is not of interest, we need just set the off-diagonal components in the matrix F (equation IV-3-8) to zero. This is the approach used in the usual re-

flectivity method.

(3) Any kind of point source, namely explosion, double couple, or other, is permitted, since the source term S in equation (IV-3-9) is left in a general form.

(4) All benefits we have discussed for our system are also applied to this new reflectivity method.

Synthetic Seismogram

Several synthetic seismograms were made to illustrate the layer reflection suppression technique by applying the integration method of chapter II. Figure 31 shows a simple test using the SCM model and an explosive source at a depth 10 km. Figure 31a is the radial component complete seismogram which is used to compare to Figure 31b, in which the free-surface reflection is 'suppressed'. As expected, Figure 31b just exhibits some easily identified arrivals such as Pn, Sn, P, PP, PS which are reflected or refracted back from the crust-mantle boundary. All other multiple phases as well as surface waves are eliminated. Figure 32 shows the similar comparison for the z component. Since a compressional type of source is used, there are no obvious direct shear signals.

For the test of reflection suppression in lower boundaries, an intermediate layer with α = 6.70 km/sec β = 3.87 km/sec, and ρ =3.0 gm/cm^3 is inserted between

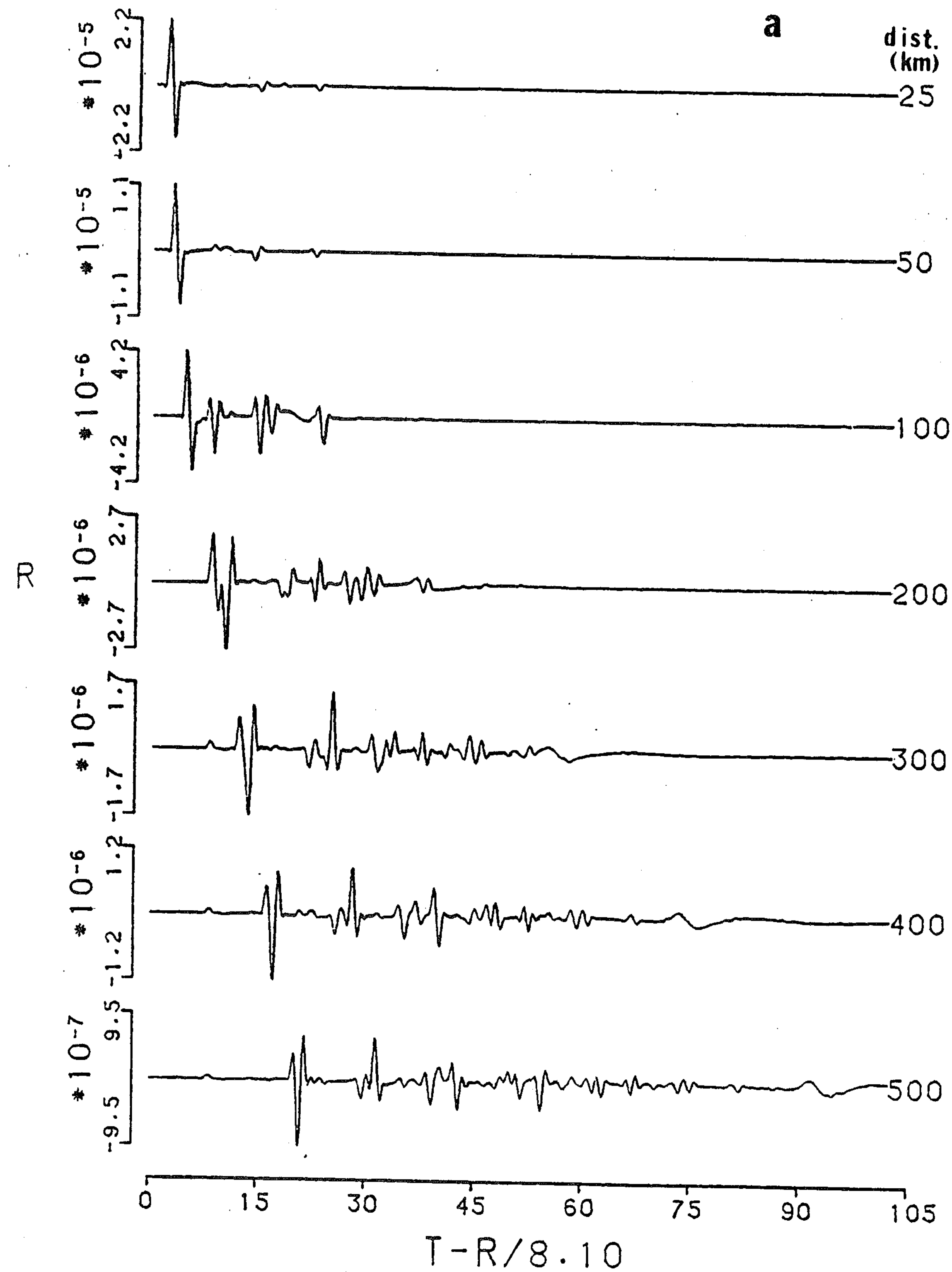


Figure 31a. Radial component complete seismograms. The SCM model and an explosive source at 10 km depth are used. A distance range of 25-500 km is presented. Multiple reflections and surface waves are well-developed, especially for large distances.

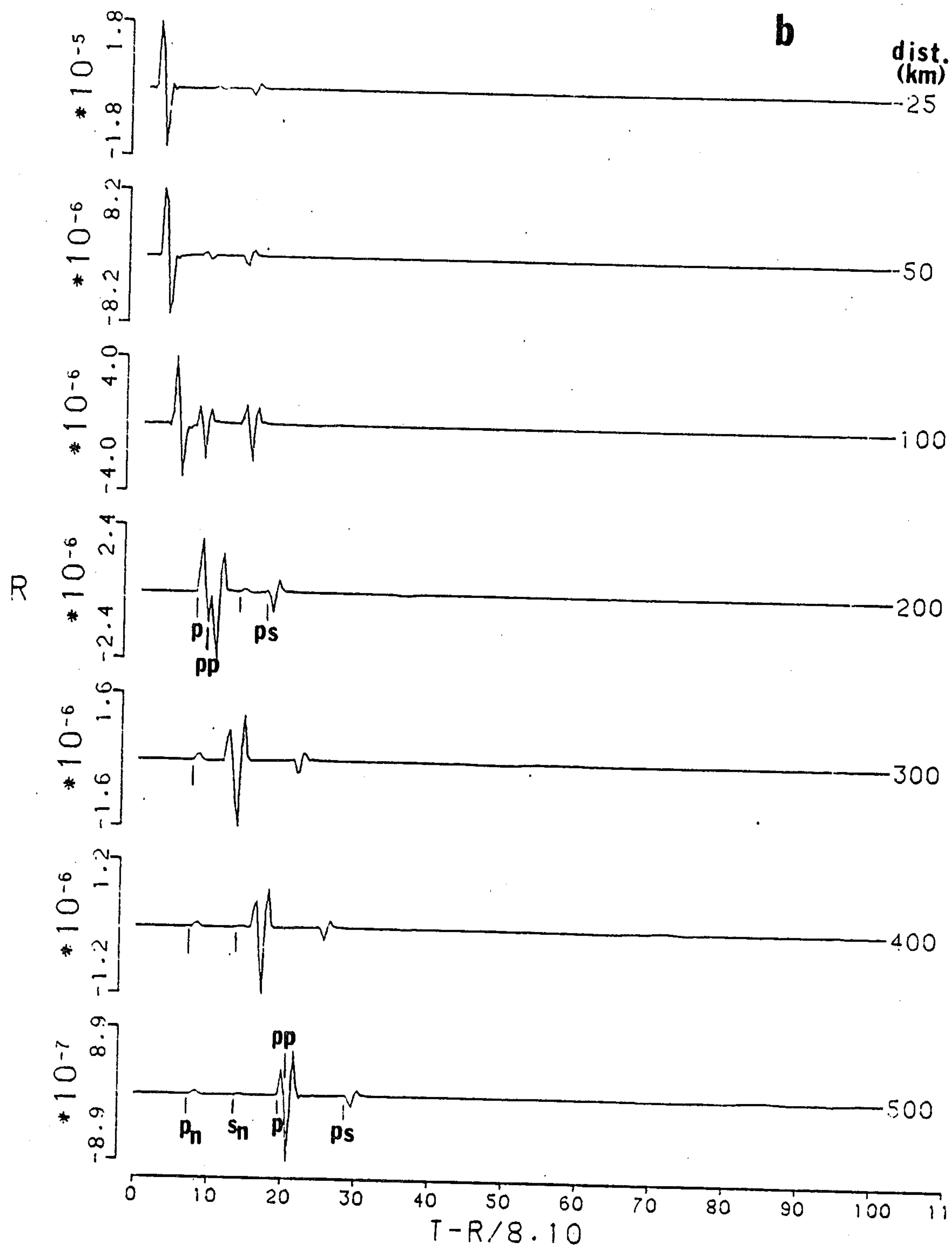


Figure 31b. Results using the same source and model of Figure 31a, but the reflectivity of the free surface is suppressed. Some identified arrivals from the crust-mantle boundary are indicated.

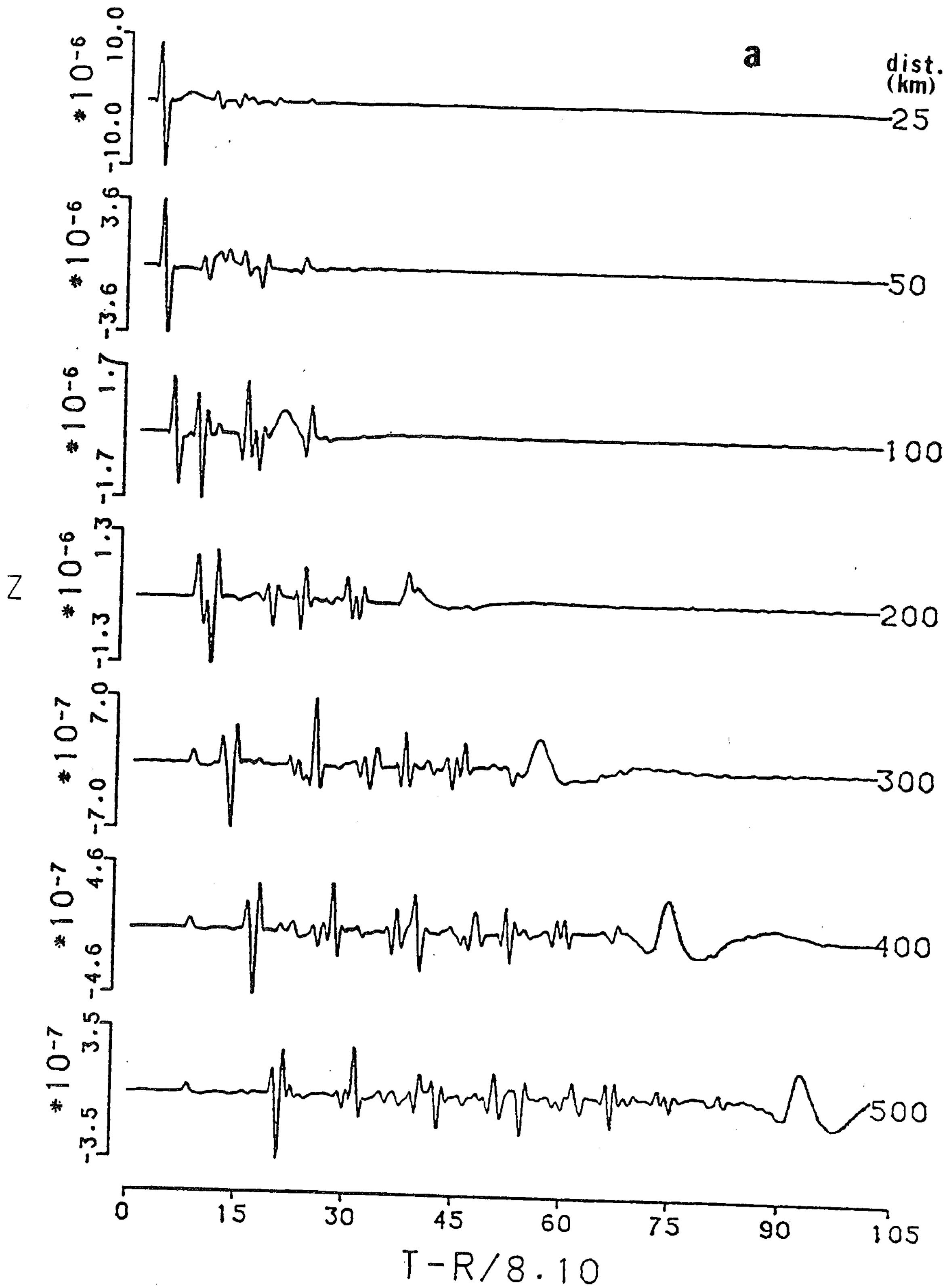


Figure 32. Results for the same source and model of Figure 31, but for the vertical component.

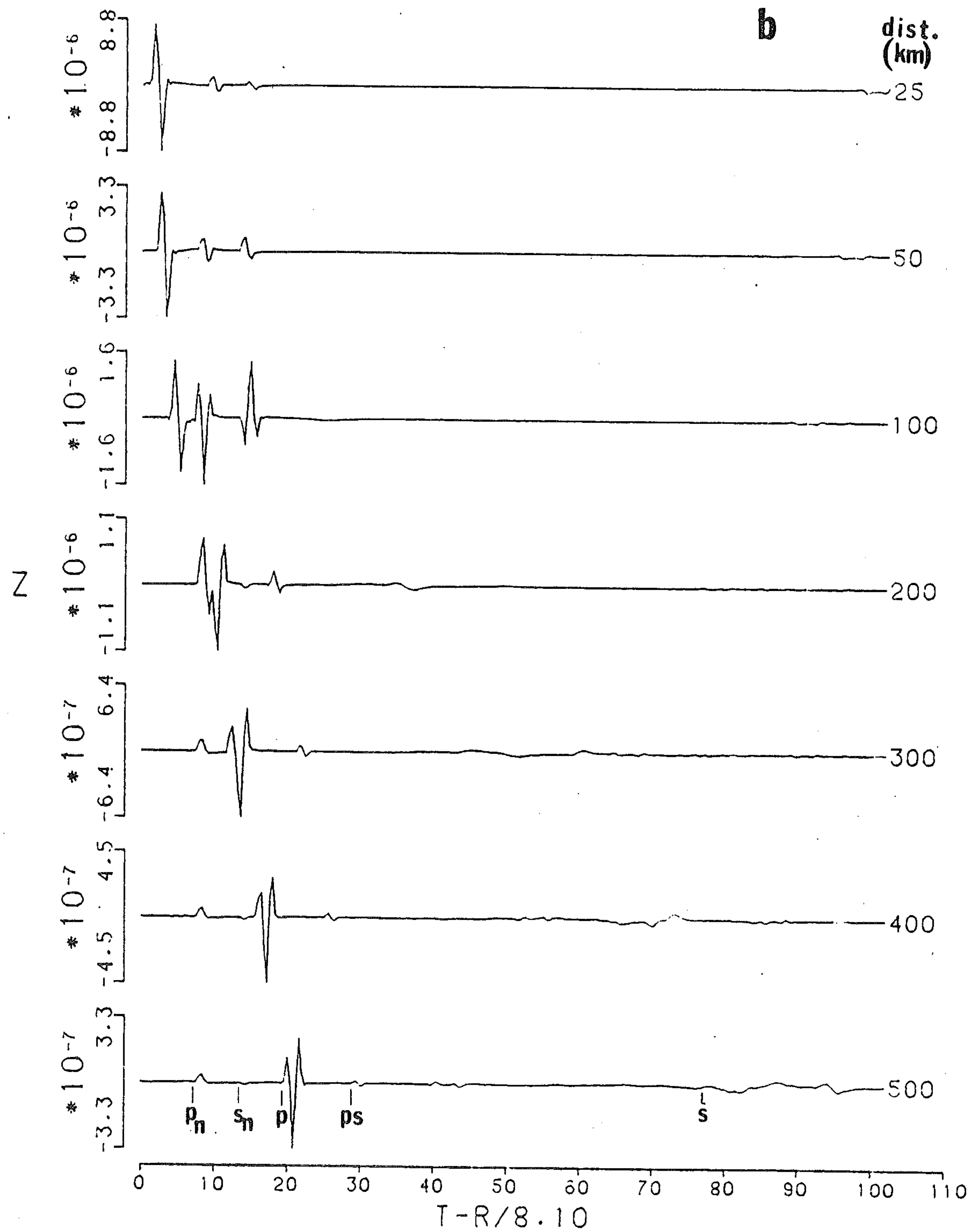


Figure 32. (cont'd)

depths 20 km and 40 Km in SCM model. Several seismograms at epicentral distances of 200 km and 300 km, due to a strike-slip source, are displayed in Figure 33, where (a) is for the SCM model, (b) is for the modified SCM model and (c) is for the modified SCM model with reflection-deprived second layer boundary. Receivers are located at an azimuth of the 0 degrees from the strike of fault. Because of the small velocity contrast across the reflection-deprived boundary, there is not much difference between (b) and (c) for the R component. However for the SH component (Figure 34), the effect is much more apparent. This means that the transverse shear wave is more sensitive to changes in reflection and transmission response than compression and SV waves. This finding might be significant for the development of shear wave exploration techniques. Seismograms (a) and (c) in Figures 33 and 34 are not comparable except for arrival time. Hence the choice of correct model in seismogram simulation is important.

If the free surface reflection of the modified SCM model is further suppressed, the seismogram from a dislocation source will be deprived of most of its wave energy, and numerical noise becomes apparent. We do not plot this result; instead, an explosive source is used for the example. Figure 35 shows the seismograms at 100, 200, 300 km epicentral distances for the modified SCM model. The top traces are complete seismograms and

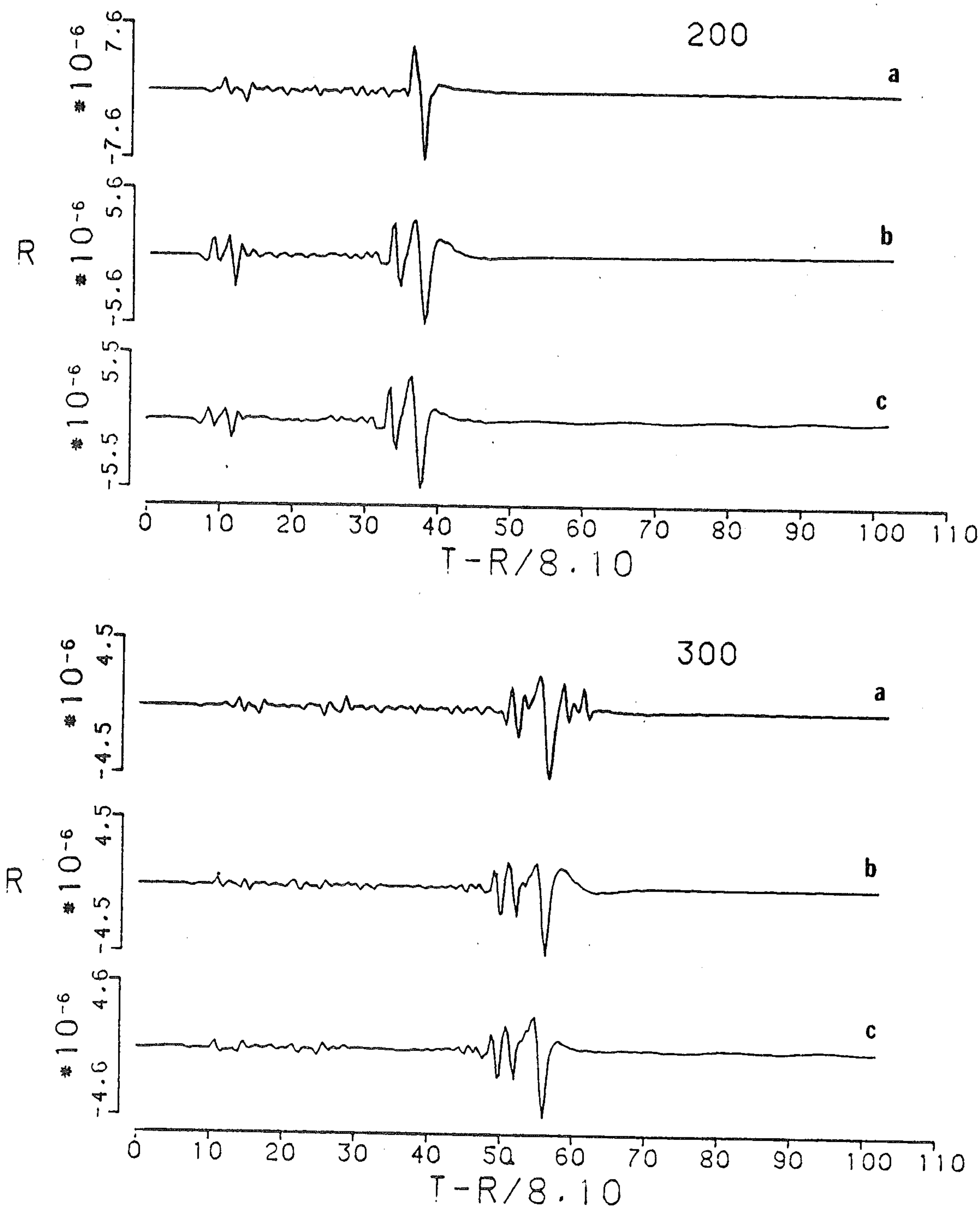


Figure 33. The effect of reflection suppression. The displays are for the radial component due to a strike-slip source buried at 10 km depth, and for stations at distances of 200 and 300 km. In each set, (a) is for the two-layer SCM model; (b) is for the three-layer modified SCM model; and (c) is for the modified SCM model with the reflections from the secondary interface suppressed.

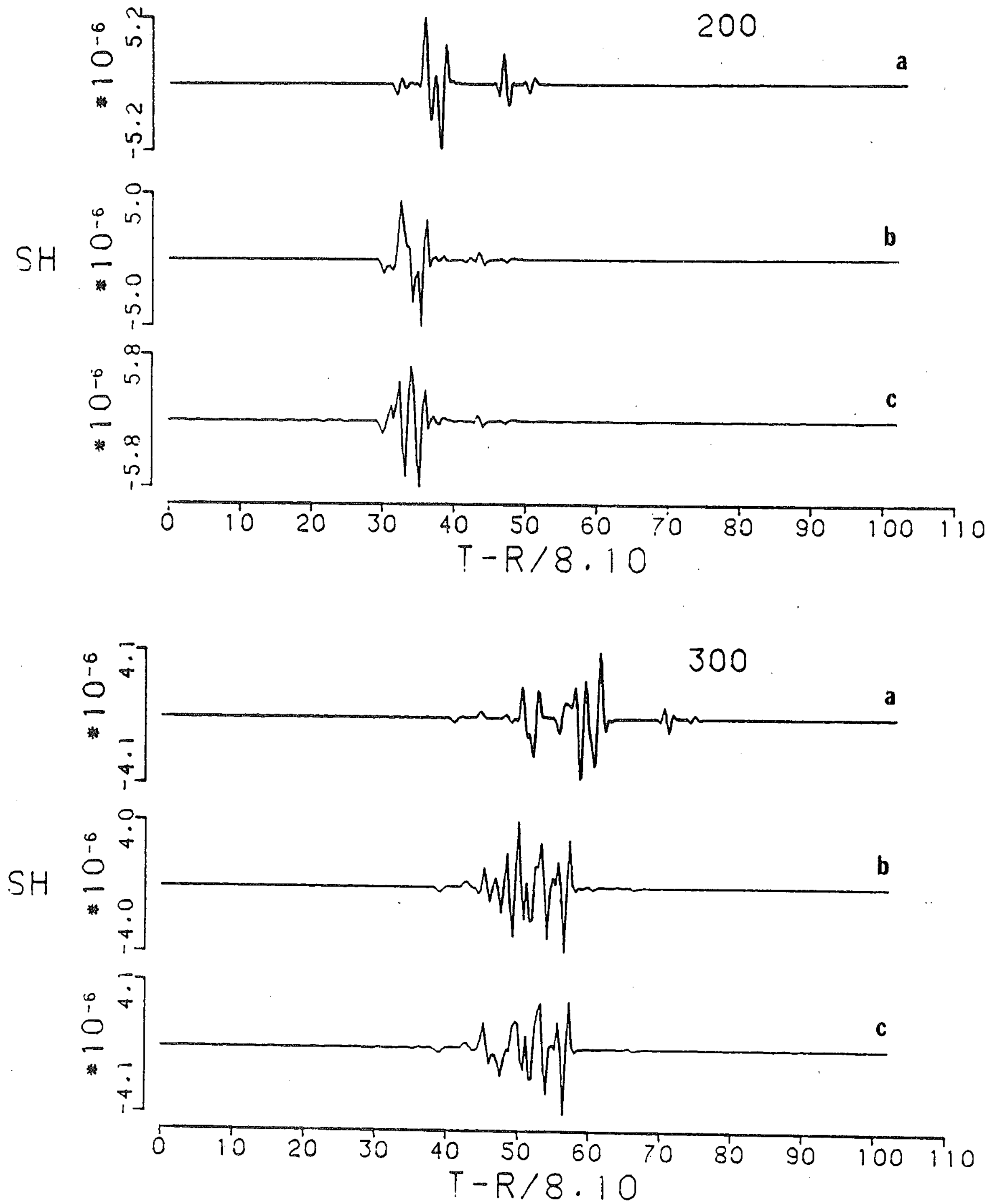


Figure 34. Results using the same source and model of Figure 33, but for the tangential component.

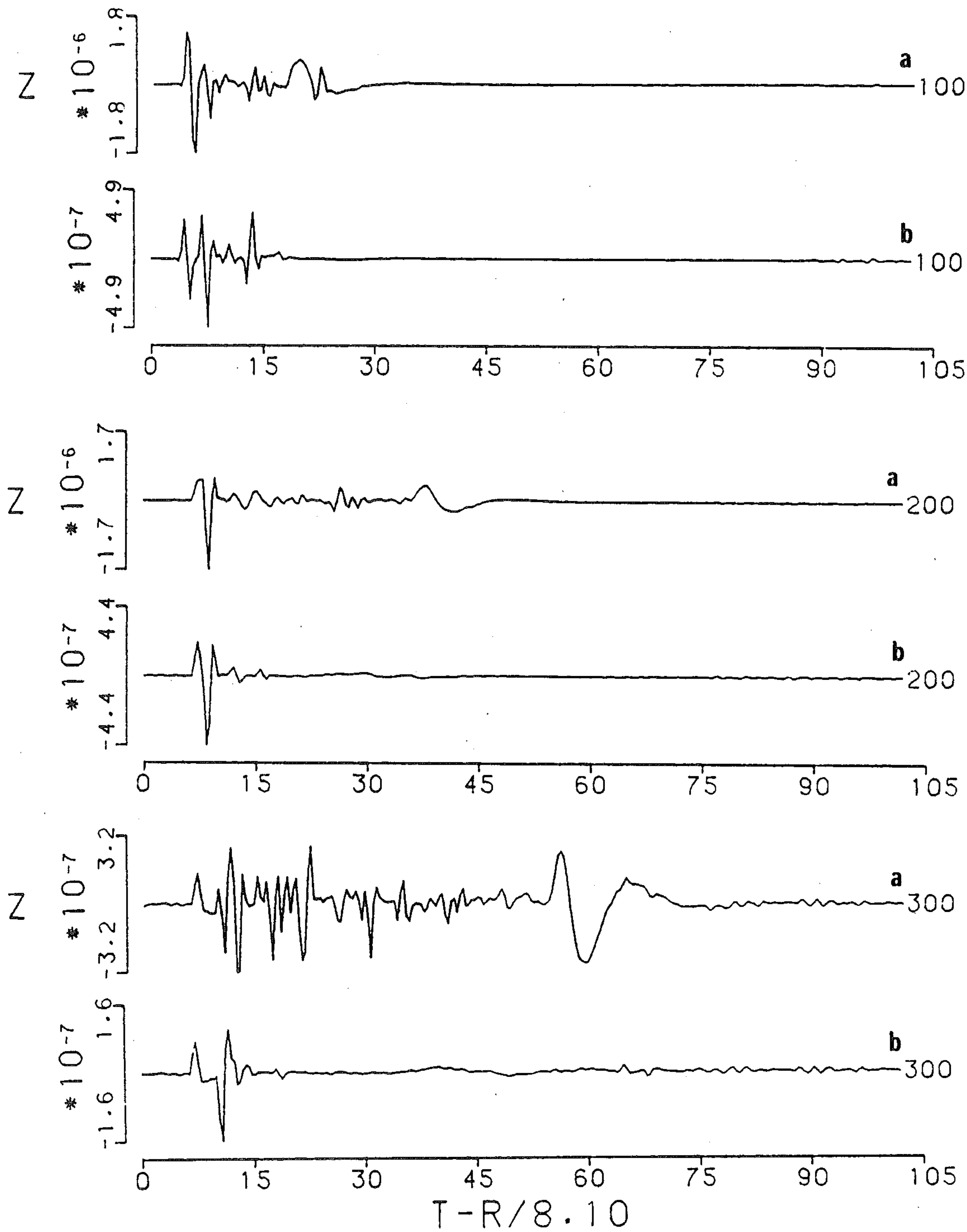


Figure 35. (a) Vertical component complete seismograms due to an explosive source buried at 10 km depth for the modified SCM model. (b) is the seismogram for the same model but with the reflections from the free surface and secondary interface suppressed. Compare (b) with Figure 32b.

the bottom traces are for reflections only from the crust-mantle boundary. The surface waves are all cancelled. Only those phases shown in Figure 32b remain.

CHAPTER V

SOURCE AND INSTRUMENT

For the completeness of the system developed in this dissertation, two factors, namely source and receiver, should also be included. The literature discussing sources is vast, and we will touch on only the major points needed to synthesize seismograms. For modeling the instrument, a method from Mitchell and Landisman (1969) is revised to more efficiently retrieve the instrument response parameters. These parameters are included in the final stage to make up the synthetic seismograms.

5.1 Source Considerations

The source factor can be included in the solutions by simply adding a singularity to the wave equation. Since the equivalence between body force expression and the discontinuity of displacement and strain at the source location was demonstrated (Burridge and Knopoff, 1964), the development of source theory has become much easier by the use of Green's functions. A Green's function is a solution of a differential equation involving the source singularity, which is independent of the form of the source function and is determined only by the differential equation and the boundary conditions. One useful application of the Green's function is that the solutions from an arbitrary source, for example a single force **f** , can be set up by convolving the Green's function and this source function:

$$u_i = G_{ij} * f_j \ .$$

If the source becomes more complicated, such as second order sources (couple or dipole), more derivatives with respect to the direction are involved:

$$u_i = - u_{i,k} \, n_k \ ,$$

where n_k represents the direction. These extra directional dependences can be properly described by a moment tensor **M** ;

$$u_i = G_{ij,k} * M_{jk} \ .$$

A detailed discussion of the concept of a moment tensor is given in Aki and Richards (1980, chapters 3 and 4). In a cylindrical coordinate system, the Green's function of the wave equation includes the kernels in three directions, i.e., r, z, and ϑ . Hence the Green's function itself also contains the directivity terms ϑ . Usually the Green's function is not easy to find. For the purpose of extracting the source dependent terms, a whole space solution from Stoke's formula (Aki and Richards 1980, equation 4.23), which has been extensively discussed in Love (1944), is expanded to a form in which it is easy to separate the Green's function and moment tensor.

One of the merits of wave integral theory is that the solutions can be represented in a suitable canonical form. That is, in a form in which the parameters of interest are taken into account at discrete points, so that the parameter input at the end of the computation chain can be varied without having to repeat the previous computations. Hence, it is desirable to factor out the directivity dependence terms, or equivalently the radiation pattern, from the source representation and calculate only some fundamental source type solutions. These solutions are combined with the directional dependences in the last stage to form the final solution. Equation (II-2-15) is such an example where three fundamental types of solutions, i.e., SS, DS, and DD, are prepared before any orientation of double couple source is considered.

Since it is not easy to isolate directivity dependences from source expressions in the forms of the moment tensor, we choose the other approach from Haskell (1963). Using a direct and clear style, Haskell (1963, 1964) listed five source types, i.e., single force, single dipole, single couple, double couple, and compression source, in terms of a discontinuity in his vector $\mathbf{K} = [A'+A'', A'-A'', B'-B'', B'+B'']^T$. A matrix F of Wang and Herrmann (1980) has been used to transform Haskell's K vector to the form $[A'', B'', A', B']^T$ and in section 2.2 we further used a matrix **E** to convert this form to a vector representing the discontinuity in the motion-stress vector (equation II-2-2). Using Σ^H to represent the terms from Haskell (1964), Σ to represent a discontinuity in potential-constant vector, and S to represent a discontinuity in motion-stress vector, there exists the following relations:

$$\mathbf{S} = \mathbf{E}\,\Sigma$$
$$\Sigma = \mathbf{F}\,\Sigma^H ,$$

where

$$\mathbf{F} = \frac{1}{2}\begin{bmatrix} -1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ -1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} .$$

These relations can be used to relate the different source expressions listed below.

Choose the coordinate system (x,y,z) as (1) x = north (2) y = east (3) z = downward into the earth, and **f** for the force acting at the source and **n** the normal to the fault plane, as defined in Haskell (1963). For an order one source, the only form is the single force. The solution can be expressed by

$$\mathbf{u} = f_3\, \mathbf{S}^0\, \mathbf{Y}_0 + (f_1 \cos\vartheta + f_2 \sin\vartheta)\, \mathbf{S}^1\, \mathbf{Y}_1$$

$$4\pi\omega^2\,\mathbf{S}^0 = [\,0,\,0,\,-2k,\,0\,]^T$$
$$4\pi\omega^2\,\mathbf{S}^1 = [\,0,\,0,\,0,\,-2k\,]^T$$

where $\mathbf{f} = [f_1, f_2, f_3]$ and ϑ is the receiver azimuth measured from the north direction clockwise. $\mathbf{Y}_n$ represents $U_z J_n$ for z component, and $U_{r,\vartheta} J_{n-1}$ for r, ϑ components as in equation (II-2-14).

For sources of order two, we find five fundamental forms are necessary, by examining the Haskell's (1964) source functions. They are

$$4\pi\omega^2\,\mathbf{S}^{0'} = [\quad 0,\quad 0,\quad 0,\quad k^2\,]^T$$
$$4\pi\omega^2\,\mathbf{S}^0 = [\quad 0,\ 2kk_\alpha^2/\rho,\quad 0,\ 4k^2\beta^2/\alpha^2\,]^T$$
$$4\pi\omega^2\,\mathbf{S}^{1'} = [\quad 0,\quad 0,\ 2k^2,\quad 0\,]^T$$
$$4\pi\omega^2\,\mathbf{S}^1 = [\,-2kk_\beta^2/\rho,\quad 0,\quad 0,\quad 0\,]^T$$
$$4\pi\omega^2\,\mathbf{S}^2 = [\quad 0,\quad 0,\quad 0,\quad k^2\,]^T\,.$$

The solutions for different source types are obtained after combining part or all of these fundamental forms. Such as,

dipole without moment:

$$\mathbf{u} = [\,(1-3f_3^2)\,\mathbf{S}^{0'} + f_3^2\,\mathbf{S}^0\,]\,\mathbf{Y}_0$$
$$+ [\,f_1 f_3 \cos\vartheta + f_2 f_3 \sin\vartheta\,]\,\mathbf{S}^1\,\mathbf{Y}_1$$
$$+ [\,(f_2^2 - f_1^2)\cos 2\vartheta - 2f_1 f_2 \sin 2\vartheta\,]\,\mathbf{S}^2\,\mathbf{Y}_2$$

single couple:

$$\mathbf{u} = [\,-3f_3 n_3\,\mathbf{S}^{0'} + f_3 n_3\,\mathbf{S}^0\,]\,\mathbf{Y}_0$$
$$+ [\,((f_1 n_3 - f_3 n_1)\cos\vartheta + (f_2 n_3 - f_3 n_2)\sin\vartheta\,)\,\mathbf{S}^{1'}$$
$$+ (f_1 n_3 \cos\vartheta + f_2 n_3 \sin\vartheta)\,\mathbf{S}^1\,]\,\mathbf{Y}_1$$

$$+ [\ (f_2n_2-f_1n_1)\cos 2\vartheta - (f_1n_2+f_2n_1)\sin 2\vartheta\ ]\ \mathbf{S}^2\,\mathbf{Y}_2$$

double couple without moment:

$$\mathbf{u} = [\ -6f_3n_3\,\mathbf{S}^{0'} + 2f_3n_3\,\mathbf{S}^0\ ]\,\mathbf{Y}_0$$
$$+ [\ (f_1n_3+f_3n_1)\cos\vartheta + (f_2n_3+f_3n_2)\sin\vartheta\ ]\ \mathbf{S}^1\,\mathbf{Y}_1$$
$$+ [\ 2(f_2n_2-f_1n_1)\cos 2\vartheta - 2(f_1n_2+f_2n_1)\sin 2\vartheta\ ]\ \mathbf{S}^2\,\mathbf{Y}_2$$

center of compression:

$$\mathbf{u} = \ \mathbf{S}^0\,\mathbf{Y}_0\ .$$

Note that for far-field solutions, $\mathbf{S}^{0'}\mathbf{Y}_0$ and $\mathbf{S}^2\mathbf{Y}_2$ differ only by a constant phase term. Hence, for far-field double-couple dislocation sources, only three independent solutions are required, and the explosion solution is just one of them.

The fundamental types as expressed in terms of Σ have the forms:

$$4\pi\omega^2\,\Sigma^{0'} = [\ k^3/2\nu_\alpha,\quad -k/2,\quad -k^3/2\nu_\alpha,\quad -k/2\ ]^T$$
$$4\pi\omega^2\,\Sigma^{0} = [\ kk_\alpha^2/\nu_\alpha,\quad 0,\quad -kk_\alpha^2/\nu_\alpha,\quad 0\ ]^T$$
$$4\pi\omega^2\,\Sigma^{1'} = [\ -k^2,\quad k^2/\nu_\beta,\quad -k^2,\quad -k^2/\nu_\beta\ ]^T$$
$$4\pi\omega^2\,\Sigma^{1} = [\ 2k^2,\quad -(\nu_\beta+k^2/\nu_\beta),\quad 2k^2,\quad (\nu_\beta+k^2/\nu_\beta)\ ]^T$$
$$4\pi\omega^2\,\Sigma^{2} = [\ k^3/2\nu_\alpha,\quad -k/2,\quad -k^3/2\nu_\alpha,\quad -k/2\ ]^T$$

for second order sources, and

$$4\pi\omega^2\,\Sigma^{0'} = [\ k,\quad -k/\nu_\beta,\quad k,\quad k/\nu_\beta\ ]^T$$
$$4\pi\omega^2\,\Sigma^{0} = [\ -k^2/\nu_\alpha,\quad 1,\quad k^2/\nu_\alpha,\quad 1\ ]^T$$

for a single-force type of source.

Similar relations can be derived for SH components. The results are

$$\mathbf{u} = [\, 2(f_1 \sin\vartheta - f_2 \cos\vartheta)\,]\, \mathbf{S}^1\, \mathbf{Y}_1$$

$$4\pi\omega^2\, \mathbf{S}^1 = [\ \ 0,\ k\,\omega^2\ ]^T$$

for a single-force source. For second order sources, the following are defined:

dipole without moment:

$$\mathbf{u} = [\, (f_2 f_3 \cos\vartheta - f_1 f_3 \sin\vartheta]\, \mathbf{S}^1\, \mathbf{Y}_1$$
$$+ [\, (f_1^2 - f_2^2) \sin 2\vartheta - 2 f_1 f_2 \cos 2\vartheta\,]\, \mathbf{S}^2\, \mathbf{Y}_2$$

single couple:

$$\mathbf{u} = [\, f_1 n_2 - f_2 n_1\,]\, \mathbf{S}^0\, \mathbf{Y}_0$$
$$+ [\, f_2 n_3 \cos\vartheta - f_1 n_3 \sin\vartheta\,]\, \mathbf{S}^1\, \mathbf{Y}_1$$
$$+ [\, (f_1 n_1 - f_2 n_2) \sin 2\vartheta - (f_1 n_2 + f_2 n_1) \cos 2\vartheta\,]\, \mathbf{S}^2\, \mathbf{Y}_2$$

double-couple without moment:

$$\mathbf{u} = [\, (f_2 n_3 + f_3 n_2) \cos\vartheta - (f_1 n_3 + f_3 n_1) \sin\vartheta\,]\, \mathbf{S}^1\, \mathbf{Y}_1$$
$$+ 2\, [\, (f_1 n_1 - f_2 n_2) \sin 2\vartheta - (f_1 n_2 + f_2 n_1) \cos 2\vartheta\,]\, \mathbf{S}^2\, \mathbf{Y}_2$$

where

$$4\pi\omega^2\, \mathbf{S}^0 = [\qquad 0,\ k^2\omega^2\ ]^T$$
$$4\pi\omega^2\, \mathbf{S}^1 = [\, 2kk_\beta^2/\rho,\qquad 0\ ]^T$$
$$4\pi\omega^2\, \mathbf{S}^2 = [\qquad 0,\ k^2\omega^2\ ]^T\ .$$

5.2 Instrument Response Parameters by Least-Square Inversion

Mitchell and Landisman (1969) described a method for determining the instrumental constants of an electromagnetic seismograph from a digitized calibration pulse. They applied a least-square technique to fit the pulse by varying four instrumental constants: period of seismometer (Ts) and galvanometer (Tg), and damping of seismometer (hs) and galvanometer (hg). A Fourier transformation was used to calculate the synthetic pulse from a known relationship between the constants in the frequency domain before comparison to the observed calibration pulse. Jarosch and Curtis (1973) simplified this method by deriving explicit equations for the pulse in the time domain, and included the scale factor as a free parameter to be determined. Mitronovas (1976) added another parameter, the original time, in the least-square fitting procedures to improve the error involved in specifying the onset time in pulse digitization.

This section will develop a similar method, but now, fitting the response parameters (to be defined in equation (V-2-1)) rather than the instrument constants Ts, hs, etc. It is important to realize that these parameters are the ones which determine the effects of the seismograph on the seismogram, not the instrument constants, although, theoretically both are mutually related. The seismograph is considered to be a 'black box'. Only linear response of the system is assumed; nothing about the instrument itself need be known. Such a method makes it possible to fit the calibration pulse of any instrument in the time domain, especially those for which suitable formula to relate the instrument constants and the system response do not exist. There are several factors influencing the results of the time domain inversion method, such as the relation of the trial values to the actual, precision of the reference pulse, and errors in digitization. If these factors are highly controlled, the new method will be of practical use especially for the short-period instruments.

Most instruments in the current seismic detecting stations usually can be considered as a linear and stable system. A well-known response function approximating this system is: (Luh, 1977)

$$bF(s) = \frac{b\, s^m}{\sum_{j=1}^{n+1} a_j s^{j-1}} \quad , \qquad \text{(V-2-1)}$$

where s is the Laplace transform variable, b the scale factor, and the a's are assigned as the instrument response parameters. b and a's are all real positive coefficients with $a_{n+1} = 1$, $a_1 \neq 0$. These parameters represent the system, and are those to be determined by the least-square inversion. m,n are chosen as integers representing the slopes near both ends of the amplitude response bands. The function to be fitted is given by

$$z(t) = bf(t, a_n, a_{n-1}, \cdots, a_1) \; ,$$

where $f(t)$ is the time domain counterpart of $F(s)$. Perturbing this function, the error will be

$$e_i = bf_i + b\sum_{j=1}^{n}\frac{\partial f_i}{\partial a_j}\delta a_j + f_i\,\delta b - z_i\ ,$$

where i indicates the data point on the time axis, and $f_i = f(t_i)$. For a least-squares fit, we want the sum of the squares of the errors or residuals (SSR) to be a minimum, i.e., $E = \sum_i e_i^2$ to be minimum and, hence,

$$\frac{\partial E}{\partial(\delta b)} = \frac{\partial E}{\partial(\delta a_j)} = 0 \qquad j = 1,\cdots,n\ .$$

Writing

$$\varepsilon_i = z_i/b - f_i$$

the normal equations are

$$\begin{bmatrix} \sum_i(\frac{\partial f_i}{\partial a_1})^2 & \cdots & \sum_i\frac{\partial f_i}{\partial a_1}\frac{\partial f_i}{\partial a_n} & \sum_i\frac{\partial f_i}{\partial a_1}f_i \\ \vdots & & \vdots & \vdots \\ \sum_i\frac{\partial f_i}{\partial a_n}\frac{\partial f_i}{\partial a_1} & \cdots & \sum_i(\frac{\partial f_i}{\partial a_n})^2 & \sum_i\frac{\partial f_i}{\partial a_n}f_i \\ \sum_i\frac{\partial f_i}{\partial a_1}f_i & \cdots & \sum_i f_i\frac{\partial f_i}{\partial a_n} & \sum_i f_i^2 \end{bmatrix} \begin{bmatrix} \delta a_1 \\ \vdots \\ \delta a_n \\ \frac{\delta b}{b} \end{bmatrix} = \begin{bmatrix} \sum_i\varepsilon_i\frac{\partial f_i}{\partial a_1} \\ \vdots \\ \sum_i\varepsilon_i\frac{\partial f_i}{\partial a_n} \\ \sum_i\varepsilon_i f_i \end{bmatrix}. \qquad \text{(V-2-2)}$$

Following Jarosch and Curtis (1973), we take $(\alpha + \delta\alpha)$ as the new value α', but use

$$b' = b\ \exp(\delta b/b) \qquad \text{(V-2-3)}$$

for the new scale factor, to force its fast convergence to exact value. However in practice, because of the linear dependence of b on the a's in the low frequency range (note that b and a's are terms of the numerator and denominator, respectively, in the response function V-2-1), the inclusion of b gives a constraint between b and a's, which sometimes causes the divergence of this method. Hence, an independent determination of b by other methods such as amplitude response analysis is recommended. The least-squares inversion is used to determine the a's only. In such a case the required modification of the above matrix equation is just to remove the n+1'th row and column.

Using the Laplace transformation pairs:

$$\frac{1}{s+a} \rightarrow e^{-at} \qquad \frac{1}{(s+a)^2} \rightarrow t\, e^{-at} , \tag{V-2-4}$$

it is easy to find $f(t)$ by factoring the response function $F(s)$ and taking the inverse Laplace transform:

$$\begin{aligned} F(s) &= \frac{s^m}{s^n + a_n s^{n-1} + \cdots + a_2 s + a_1} \\ &= \frac{c_1}{s-d_1} + \frac{c_2}{s-d_2} + \cdots + \frac{c_n}{s-d_n} \\ \rightarrow f(t) &= c_1 e^{d_1 t} + c_2 e^{d_2 t} + \cdots + c_n e^{d_n t} = \sum_{i=1}^{n} c_i e^{d_i t} , \end{aligned} \tag{V-2-5}$$

where

$$c_i = \frac{d_i^m}{\prod_{k=1\, k\neq i}^{n} (d_i - d_k)} .$$

The roots d_i and coefficients c_i might be complex numbers. It is noted that in this expansion no double roots are permitted. These double roots might happen when parts of the instrument are set at critical damping. However, we find that a small coupling $\sigma^2 > 0.0001$ will push the imaginary part of the roots to be nonzero. Since the roots are always in conjugate pairs, the values of the c's are determined by $Re(d_i)$ and $|d_i|$. Hence, a small disturbance in the imaginary part, which can be added arbitrarily or with the inclusion of coupling factors, will not affect the result much, but will prevent the problem of a double root.

The time functions, $\frac{\partial f(t)}{\partial a_j}$ are more difficult to find. To avoid using numerical differencing, an analytic form can be derived as follows:

$$\frac{\partial F(s)}{\partial a_j} = \frac{(-1)\, s^{j-1+m}}{(s^n + a_n s^{n-1} + \cdots + a_2 s + a_1)^2}$$

$$= \frac{(-1)\, s^{j-1+m}}{(s-d_1)^2 (s-d_2)^2 \cdots (s-d_n)^2}$$

$$= -\left[\frac{p_{1j}}{(s-d_1)^2} + \frac{q_{1j}}{s-d_1} + \frac{p_{2j}}{(s-d_2)^2} + \frac{q_{2j}}{s-d_2} + \cdots + \frac{p_{nj}}{(s-d_n)^2} + \frac{q_{nj}}{s-d_n}\right]$$

where

$$p_{ij} = \frac{d_i^{j-1+m}}{\prod_{k=1,\, k\neq i}^{n} (d_i - d_k)^2} = d_i^{j-1-m}\, c_i^2 \qquad i,j = 1,\ldots,n\,.$$

The expression for q_{ij} needs more derivations. Set

$$H(s) = \prod_{k=1\, k\neq i}^{n} (s-d_k)^2$$

q_{ij} can be evaluated by

$$q_{ij} = [\frac{s^{j-1+m}}{(s-d_i)\,H(s)} - \frac{p_{ij}}{(s-d_i)}]|_{s\to d_i}$$

$$= \frac{1}{(s-d_i)}[\frac{s^{j-1+m}}{H(s)} - \frac{d_i^{j-1+m}}{H(d_i)}]|_{s\to d_i}$$

$$= \frac{\partial}{\partial s}[\frac{s^{j-1+m}}{H(s)}]|_{s\to d_i}$$

$$= (j-1+m)\frac{s^{j-2+m}}{H(s)} - 2\frac{s^{j-1+m}}{H(s)}(\sum_{k=1\, k\neq i}^{n}\frac{1}{s-d_k})|_{s=d_i}$$

$$= p_{ij}\,[\,(j-1+m)\,d_i^{-1} - 2(\sum_{k=1\, k\neq i}^{n}\frac{1}{d_i-d_k})]$$

Hence from the Laplace transform (V-2-4), the time domain derivative $\frac{\partial f(t)}{\partial a_j}$ has the form:

$$\frac{\partial f(t)}{\partial a_j} = -\sum_{i=1}^{n}(p_{ij}te^{d_i t} + q_{ij}e^{d_i t})$$

where

$$p_{ij} = d_i^{j-1-m}\,c_i^2$$

$$q_{ij} = p_{ij}\,[(j-1+m)\,d_i^{-1} - 2\sum_{k=1\, k\neq i}^{n}\frac{1}{d_i-d_k}]$$

$$d_1^{-1} = 0$$

We find that the derivatives of $f(t)$ have the close forms in the time domain. This is difficult to attain by varying the instrumental constants, as in common least-square calibration pulse inversion methods.

Since the differences in order of values of elements in the inversion matrix (equation V-2-2) are usually about ten to twenty, a scaling constant sometimes is needed to prevent numerical overflow or underflow. It is made by simply changing the transformation factor s to $\rho s'$ and the parameters found will be scale adjusted $b' = \rho^{m-n} b$, $a_j' = \rho^{j-n} a_j$. The time axis in this case is reduced by $1/\rho$ because of the corresponding scale change in the frequency.

To test the new method, an example from Mitronovas (1976) is chosen. The results are shown in Table 2. We first compute a synthetic pulse by the equation (e) of Jarosch and Curtis (1973) for the instrument with seismometer overdamped and galvanometer underdamped, then normalize to the maximum values indicated, and round off to nearest integer values to simulate differences in the digitization accuracy. A set of a values of the instrument parameters are then obtained using the new inversion technique. Table 2 also lists the values using the method of Mitchell and Landisman (1969) as a comparison. There does not seem to be any significant difference between these methods, although the new method seems a little better in accuracy for such a synthetic case. This test confirms the success of the new method.

Figures 36 and 37 show the application to real data for the LPZ WWSSN instrument at FVM station. Figure 36 displays the digitized calibration pulse and its amplitude spectrum. Using this pulse as an input, we

TABLE 2

Comparison of Parameters for Synthetic Pulses*

Maximum Amplitude	$a_1(10^1)$	a_2	$a_3(10^{-1})$	$a_4(10^{-3})$	SSR@
100	.1066312	.2915167	.2415525	.7110602	9.60
	.1066272	.2915026	.2415318	.7110564	9.06
200	.1060013	.2893678	.2398869	.7069588	11.95
	.1059998	.2893564	.2398676	.7069502	11.94
400	.1035897	.2840534	.2350874	.6935693	17.25
	.1035844	.2840341	.2350574	.6935680	17.22
800	.1031482	.2830225	.2343075	.6912319	14.92
	.1031514	.2830207	.2342968	.6912327	14.80
8000	.1034220	.2835561	.2347712	.6924602	27.77
	.1034212	.2835407	.2347397	.6924603	16.30

*In each case the upper values are solutions based on the method of Mitchell and Landisman (1969), and the lower values are solutions based on the present method.

@Sum of squares of residuals between the reference and the calculated pulse.

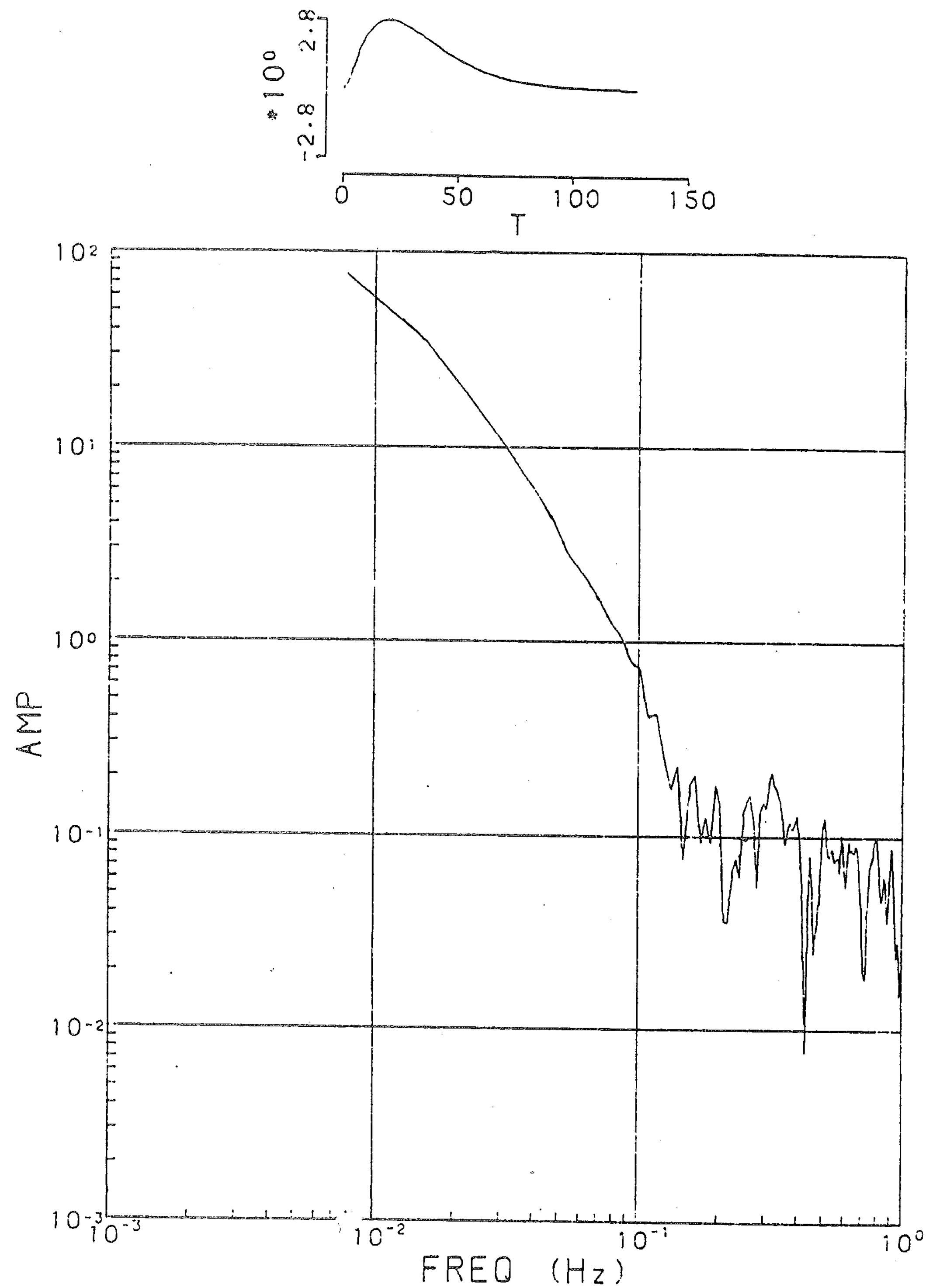


Figure 36. The calibration pulse and its amplitude spectrum for the WWSSN LPZ instrument at the station FVM.

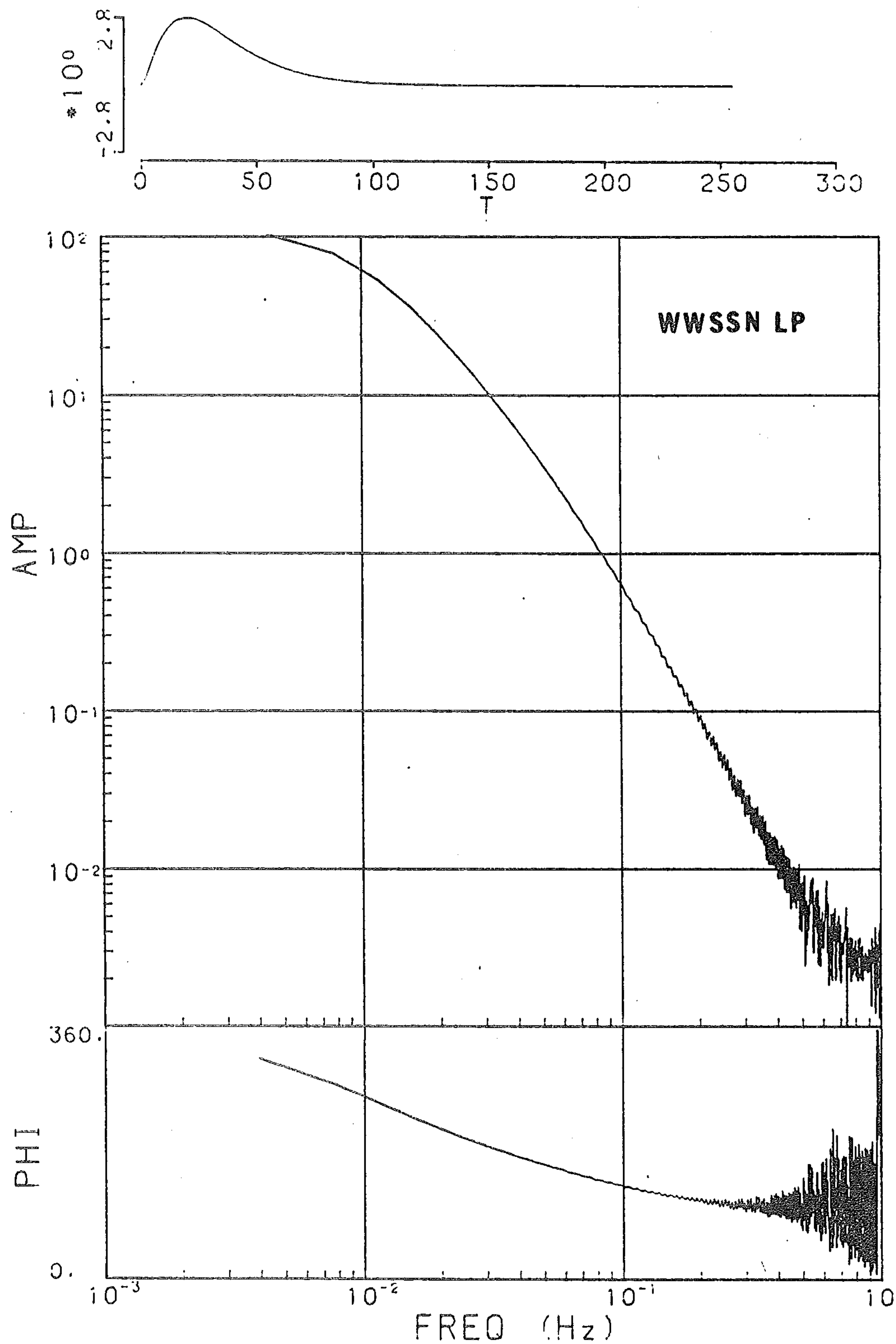


Figure 37. The impulse-response pulse and its amplitude and phase spectra of a simulated instrument obtained by applying the inversion technique to the calibration pulse of Figure 36.

determine the instrument response parameters by the present inversion method. Figure 37 shows the impulse response pulse of this simulating instrument, and its amplitude and phase responses. It is found that, except for high frequency noise, the new method can adequately model an instrument even if we lack knowledge of the instrument constants. Furthermore, the α parameters obtained can be easily incorporated in the program to describe the instrument effect. In the next section we will discuss such an application in the time domain.

5.3 Simulating the Instrument by IIR

With the response parameters determined, we can impose the instrument response (V-2-1) upon the input data by direct multiplication in the frequency domain, and then take an inverse FFT to synthesize the time series. However, if the generated seismogram is already a time history, a time domain operator representing the instrument would be more convenient to use. In this section an infinite impulse response filter (IIR) will be designed to make the instrument response superposition in the time domain by a recursive operation. The basic theory will be the classical Z transformation which has long been used to treat digital signals (Robinson and Treitel, 1980).

First let us consider one of the filters in equation (V-2-5):

$$F(s) = \frac{1}{s-d} .$$

The relationships among Fourier, Laplace, and Z transformations for this filter are easy to find,

$$\begin{aligned} e^{dt}U(t) &\rightarrow \frac{1}{i\omega - d} \\ &\rightarrow \frac{1}{s-d} \\ &\rightarrow \frac{1}{1-e^{d\Delta t}Z^{-1}} , \end{aligned} \tag{V-3-1}$$

where $U(t)$ is the Heaviside step function. Because of the causality of the system, the poles in equation (V-3-1) should be located in the left half of the complex s-plane (Papoulis, 1962), or equivalently, inside the unit circle of the complex Z-plane. If we use the following definition for the Z-transform:

$$F(Z) = \sum_{n=0}^{\infty} f(n) Z^{-n}$$

the two-term filter will be stable for such a system, since if expanded in an infinite polynomial of Z^{-1}, the coefficients are all bounded. This stable property is necessary for any IIR to simulate the instrument response. Before we design such a filter, we must be certain to check that all of the poles of the Laplace transformation have a negative real part.

From equation (V-2-5), the instrument response can be written as

$$F(s) = \sum_{i=1}^{n} \frac{c_i}{s - d_i} . \qquad \text{(V-3-2)}$$

The corresponding Z-transform will be

$$F(Z) = \sum_{i=1}^{n} \frac{c_i}{1 - e^{d_i \Delta t} Z^{-1}} . \qquad \text{(V-3-3)}$$

This formula represents n number of two-term filters operating in parallel. We are not going to sum them up as a single filter (Seidl, 1980)

$$F(Z) = \frac{a_0 + a_1 Z^{-1} + \cdots + a_{n-1} Z^{-(n-1)}}{1 + b_1 Z^{-1} + \cdots + b_n Z^{-n}} ,$$

because the numerical error makes the determination of a's and b's unsatisfactory. Besides, this form is not easy to apply (Kulhanek, 1979).

It is known that the multiplication of Z^{-1} means a shift of the time sequence by one sampling period. Applying the filter (V-3-3) to the input data, x_k , we have the output y_k ;

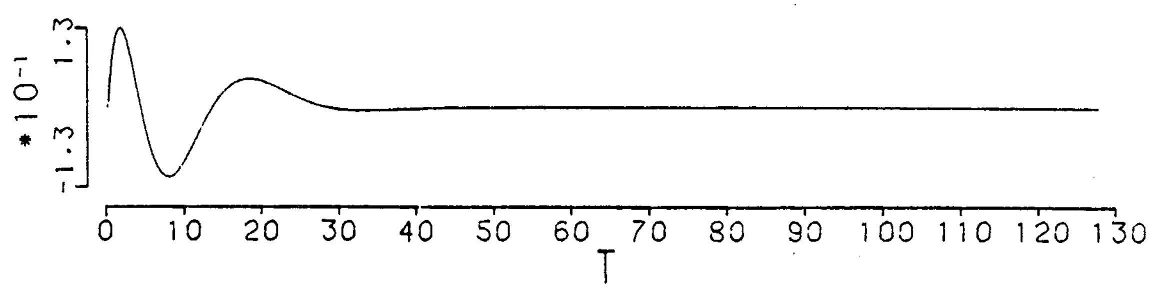
$$\begin{aligned} y_k &= F(Z)\, x_k \\ &= \left(\sum_{i=1}^{n} \frac{c_i}{1 - e^{d_i \Delta t} Z^{-1}} \right) x_k \\ &= \sum_{i=1}^{n} \left(\frac{c_i}{1 - e^{d_i \Delta t} Z^{-1}} x_k \right) \\ &= \sum_{i=1}^{n} y_k^{(i)} \end{aligned}$$

where

$$y_k^{(i)} = c_i x_k + e^{d_i \Delta t} y_{k-1}^{(i)} .$$

k is the sampling point of the sequence, and i represents a different two-term recursive filter. This is the recursive formula describing the time domain operation.

Figure 38 gives the impulse response pulse and its spectrum for an LRSM 6284-13 seismograph. The impulse response waveform is obtained by passing an impulse signal through a Z-transform IIR filter representing the instrument. Different amplitude response curves come from different time sampling intervals, which are 1.0, 0.5, and 0.0625 second for curves from top to bottom. A theoretical response curve, which falls in the position of the curve with $\Delta t = 0.0625$, is also shown for comparison. Some restriction should be considered before using recursive filters. For the Fourier transformation, the sampling along the frequency axis, Δf, gives the periodicity in the time domain, and causes the aliasing effect of the time series (Brigham, 1974). Equivalently, the time sampling Δt of the Z-transform causes the aliasing effect, but now in the frequency domain. Figure 38 shows the effect of aliasing at the ends of the frequency response curves for different sampling rates. Naturally smaller Δt values give better results. The reason for such an effect comes from $Z = e^{i\omega\Delta t}$ which is a harmonic function with a period of 2π . To keep $\omega\Delta t = 2\pi$ smaller Δt gives higher Nyquist frequency, i.e., shifts the aliasing to



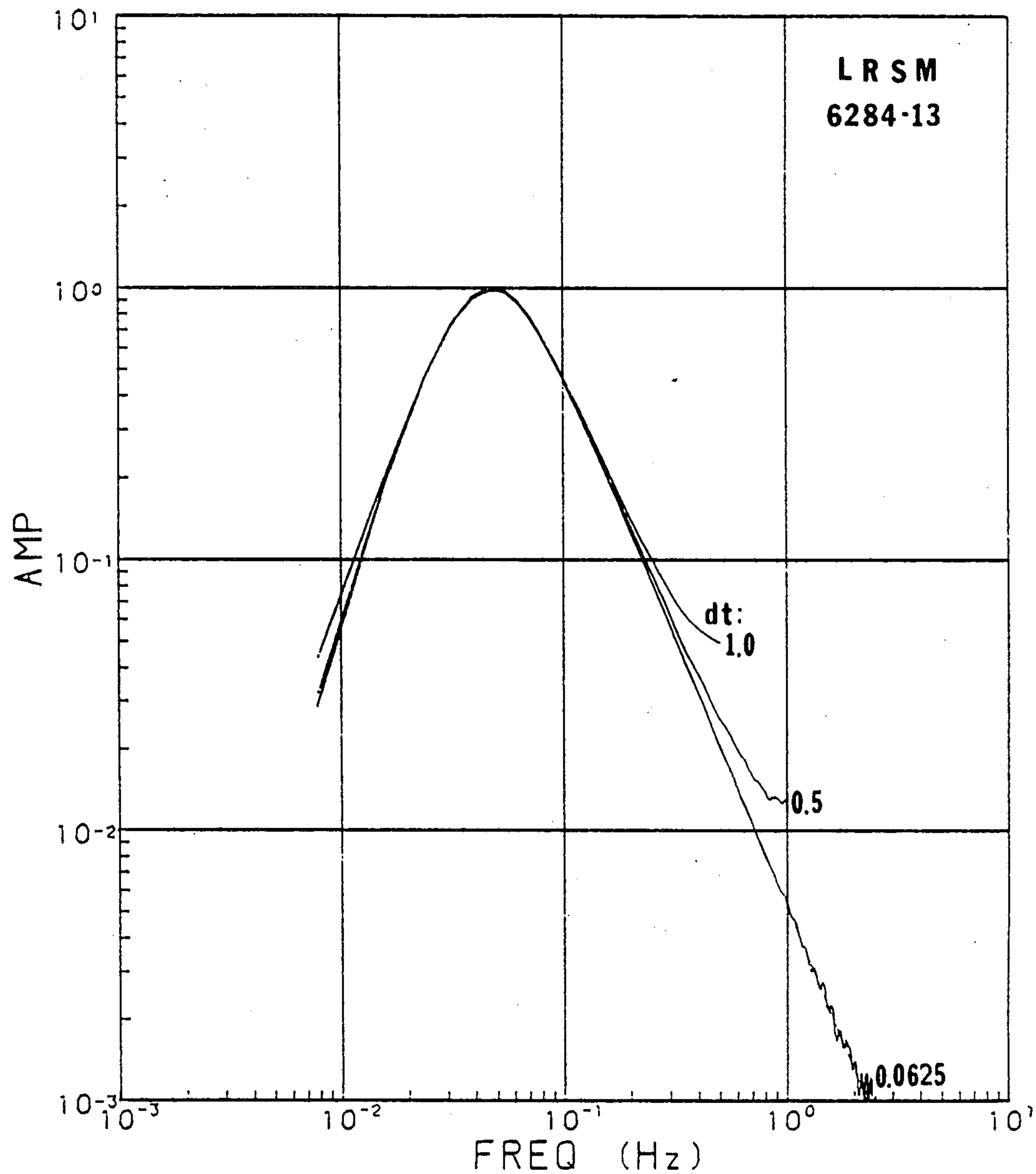


Figure 38. Simulation of an LRSM 6284-13 seismograph using the Z-transform method. Three response curves correspond to different sampling rates dt of 1.0, 0.5 and 0.0625 sec.

high frequencies.

Another way to alleviate the aliasing problem is to use the bilinear Z-transform. By substituting

$$s = \frac{2}{\Delta t}\frac{1 - Z^{-1}}{1 + Z^{-1}}$$

in equation (V-3-2), it is easy to find the recursive relation of a two-term filter for the bilinear Z-transform:

$$y_k^{(i)} = \frac{c_i}{2/\Delta t - d_i}(x_k + x_{k-1}) + \frac{2/\Delta t + d_i}{2/\Delta t - d_i}\, y_{k-1}^{(i)}$$

The bilinear Z-transform is a low frequency approximation (Oppenheim and Schafer, 1975, p.208). The distortion of the frequency axis at high frequencies offsets the true response. As shown in Figure 39, which describes the response of a short period WWSSN instrument, the bilinear Z-transform simulates the response well at low frequencies, but is poor for high frequencies. To obtain good results, a small Δt is again required. On the other hand, the filter response is found to be zero at the Nyquist frequency, which avoids any Gibb's phenomina in the time domain. A special frequency warping is used in practice to design the digital filter from its analytic form.

Using the time domain operation, we designed several IIR filters to model some currently used

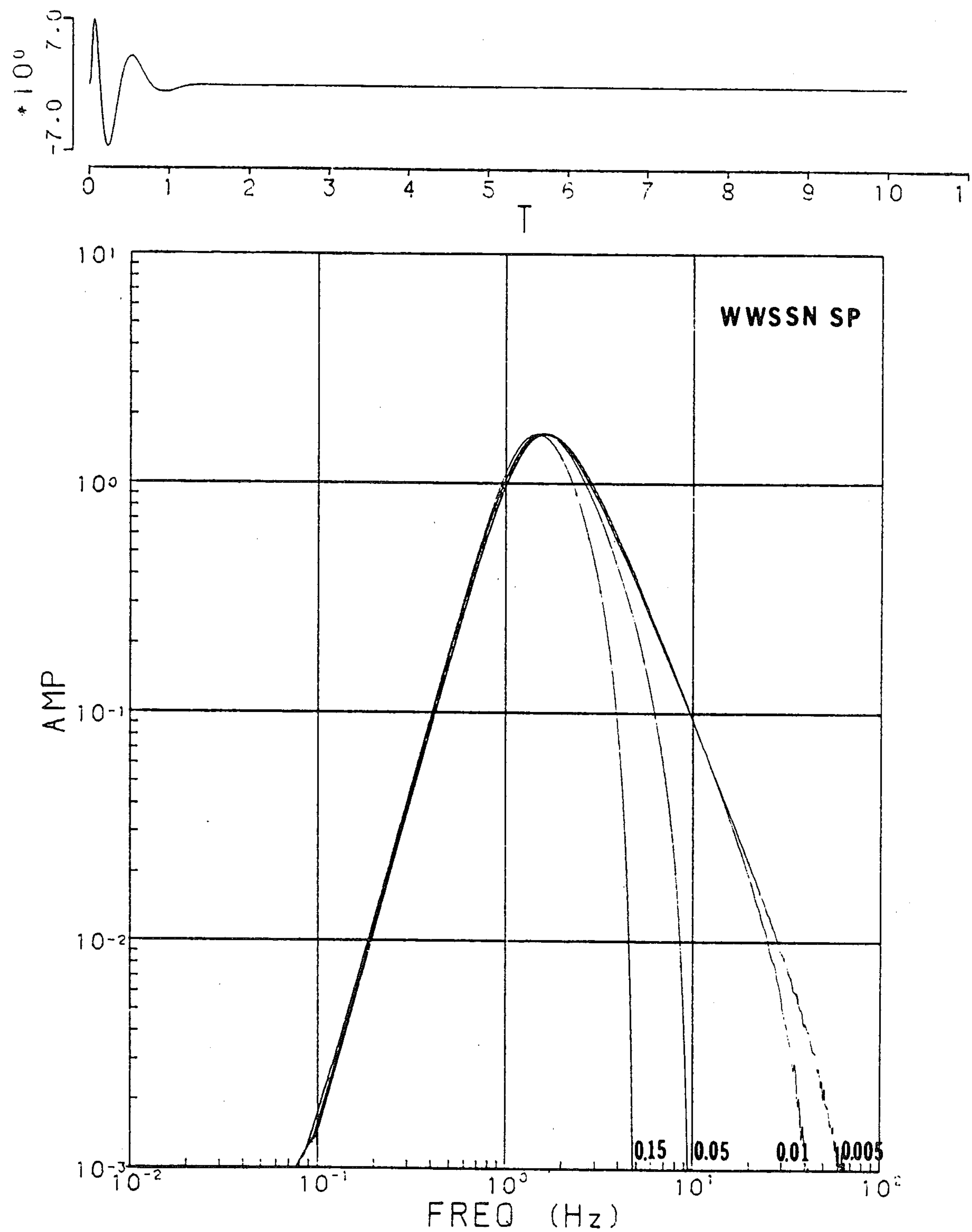


Figure 39. Simulation of a WWSSN SP instrument using the bilinear Z-transform method. Different response curves correspond to different sampling rates of 0.15, 0.05, 0.01 and 0.005 second.

instruments. Figure 40 illustrates one of the results. The seismograms plotted correspond to the time histories before and after the addition of a WWSSN SP instrument for two cases. The instrument response curve is the one given in Figure 39. It can be seen that the waveforms are totally altered.Low frequency signals are filtered out, and the remaining waveforms represent more likely the high frequency, higher mode Lg phases.

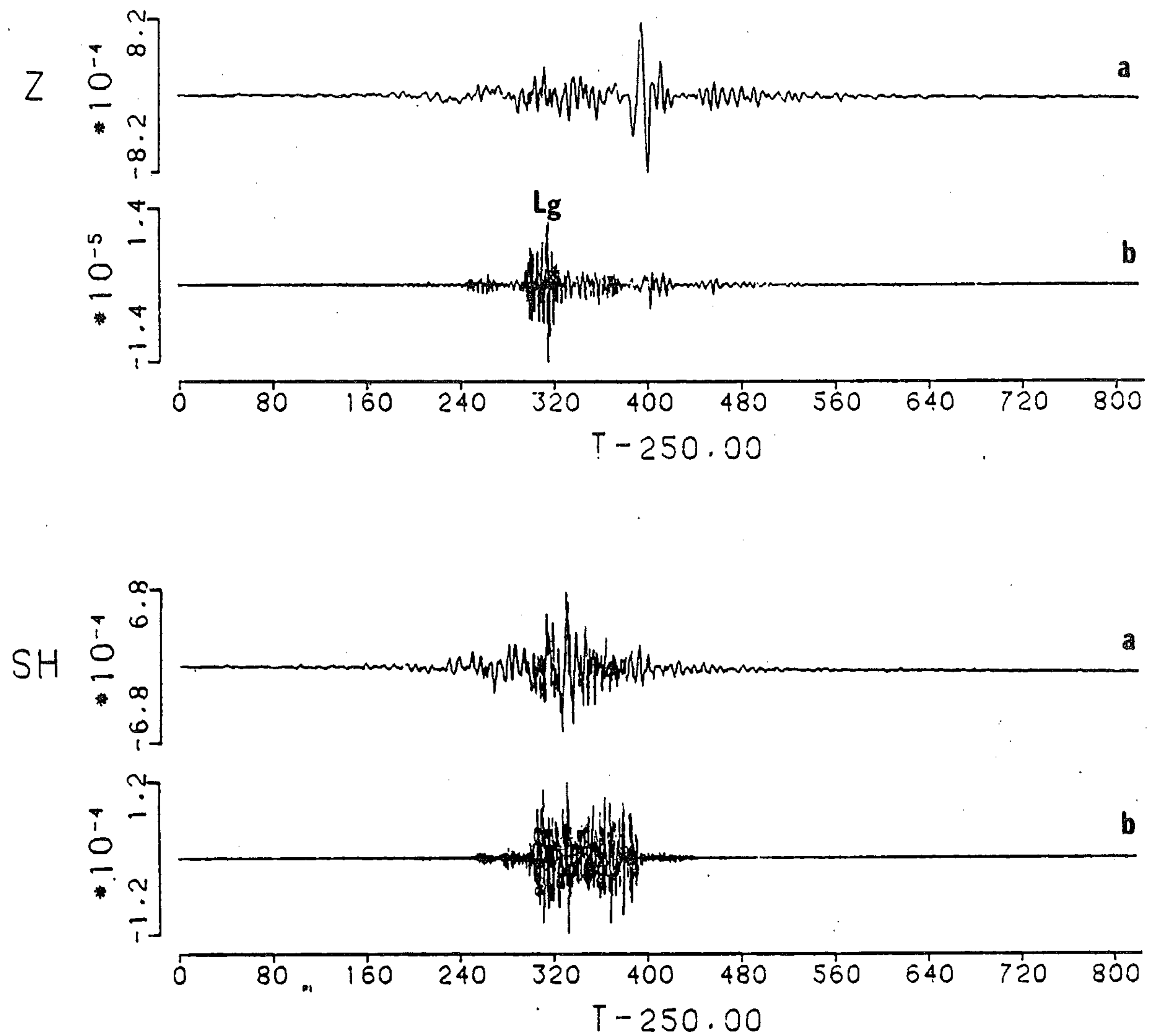


Figure 40. Seismograms showing the effect of the instrument. The two displays are for the vertical and tangential components, respectively. The CUS model and a distance of 2000 km are used. (a) is the ground displacement; and (b) is the seismogram after passing through a short-period instrument. The instrument response is that shown in Figure 39.

CHAPTER VI

COMPARISONS

To employ the synthetic seismogram method in actual applications, the validity of theory and numerical procedures should be checked. In this chapter, several comparisons of the wave integral method with totally different methods are presented. These examples illustrate the flexibility and reliability of the present method, and lend confidence to the new system.

Comparison to Generalized Ray Theory

First, the free surface displacements due to a double-couple source in a uniform half-space generated by the integral method are checked against the complete solutions from the Cagniard-de Hoop method (Johnson, 1974). Figures 41 and 42 show the computed radial components of ground velocity time histories at distances of 10, 25, 50, and 75 km for the vertical strike-slip and vertical dip-slip sources, respectively. Sections a in these figures come from a Cagniard-de Hoop ray summation and sections b from the integral method. The velocity structure has P velocity = 6.15 km/sec, S velocity = 3.55 km/sec, density = 2.8 gm/cm^3. The depth of the point dislocation is 10 km. A seismic moment of

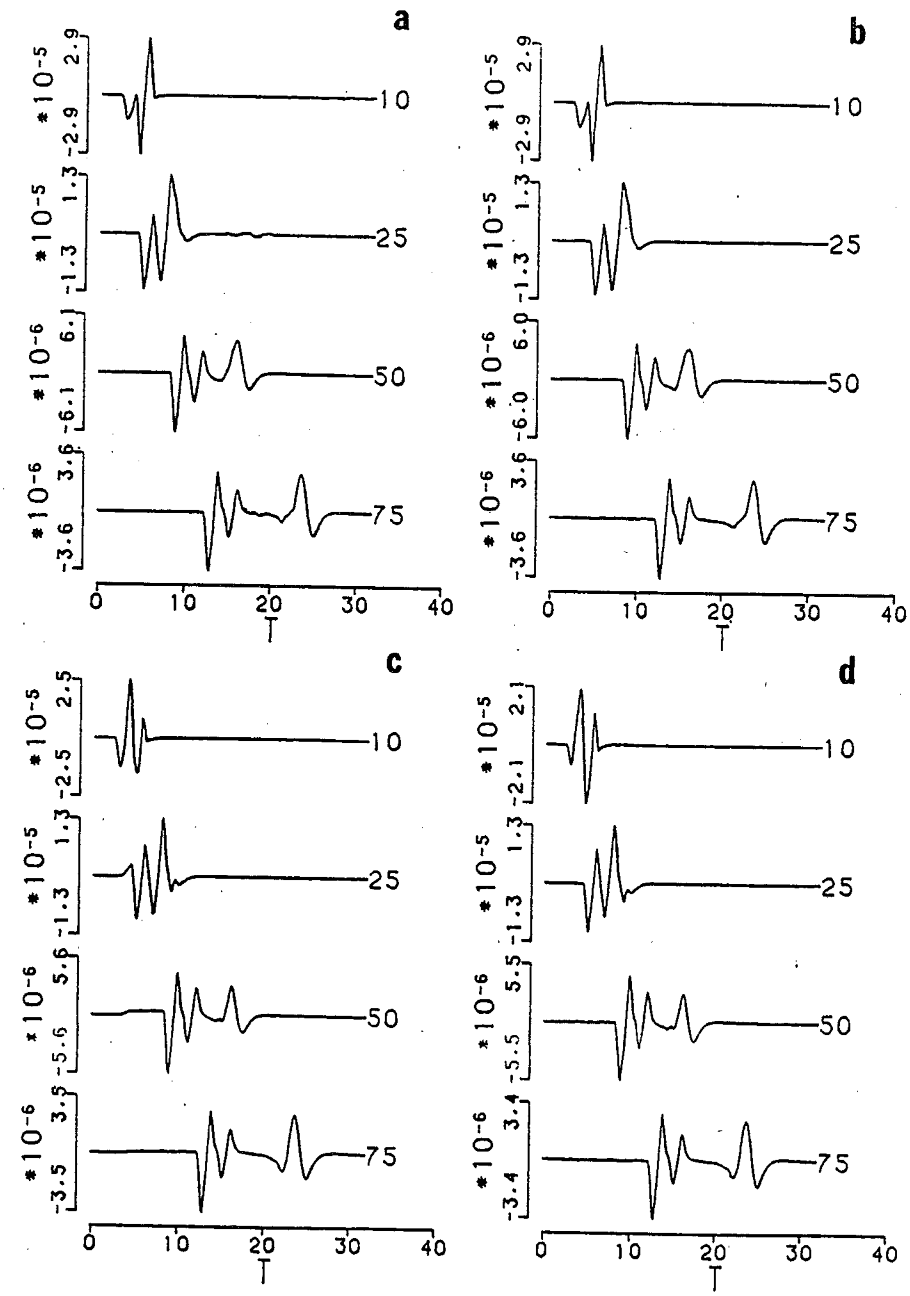


Figure 41. Comparison of Cagniard-de Hoop and wave theory solutions for a vertical strike-slip source at a depth of 10 km in a half-space with parameters of the first layer of SCM model. (a) Cagniard-de Hoop solution. (b) The complete wave theory solution. (c) The wave theory solution containing only the near-field and far-field P-SV terms. (d) The wave theory solution including only the far-field P-SV term.

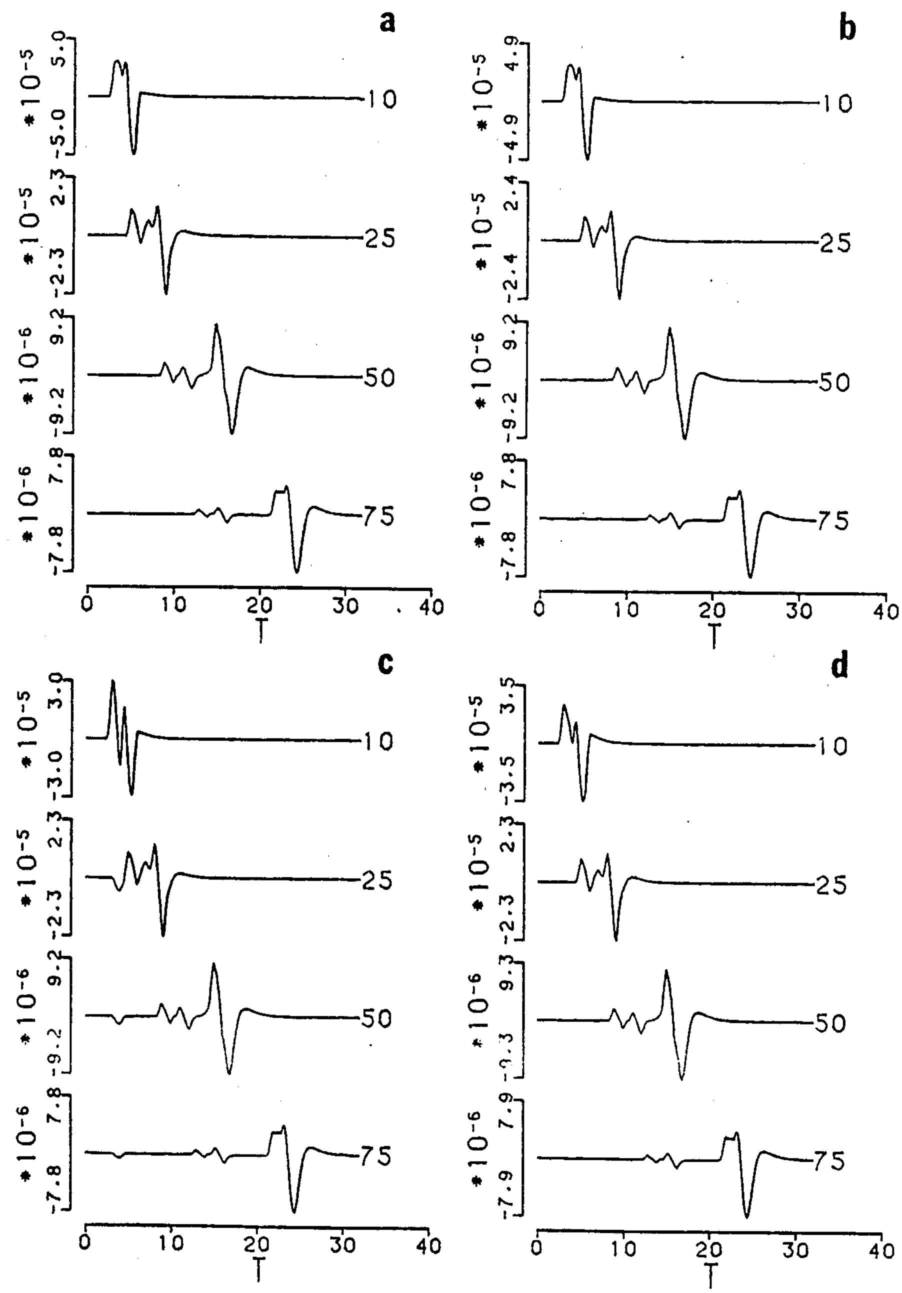


Figure 42. Results for the same model of Figure 41, but for the radial component due to a vertical dip-slip source.

1.0 E +20 dyne-cm and a parabolic source time function with τ = 0.5 sec are applied.

The agreement between the records of these two methods is excellent. The comparisons are of near-perfect precision for both fault slip prescriptions. This convinces us that the method of wave integrals is highly reliable. Figures 41 and 42 also show the effect of near-field terms (equation II-2-14) on the complete solutions. Sections c in these figures include the contributions of far-field and near-field P-SV terms (first two terms in equation II-2-14), where a noncausal, nonpropagating arrival exists as indicated by Herrmann (1978a), especially for the vertical dip-slip source. The records in sections d correspond to the solution using only the far-field term. From these figures, we can see that an apparent Rayleigh phase (fundamental mode only for half-space model) emerges from the P phase group as the distance increases. Since a clear distinction between the S arrival and surface wave phases is not possible, the separation between these two main phases on seismograms is not obvious.

Comparison to Finite-Element Method

The second example (Figure 43, 44) shows the accuracy of the present solution for a horizontally layered media. The compared solutions are obtained using the finite element method, which are adopted directly from

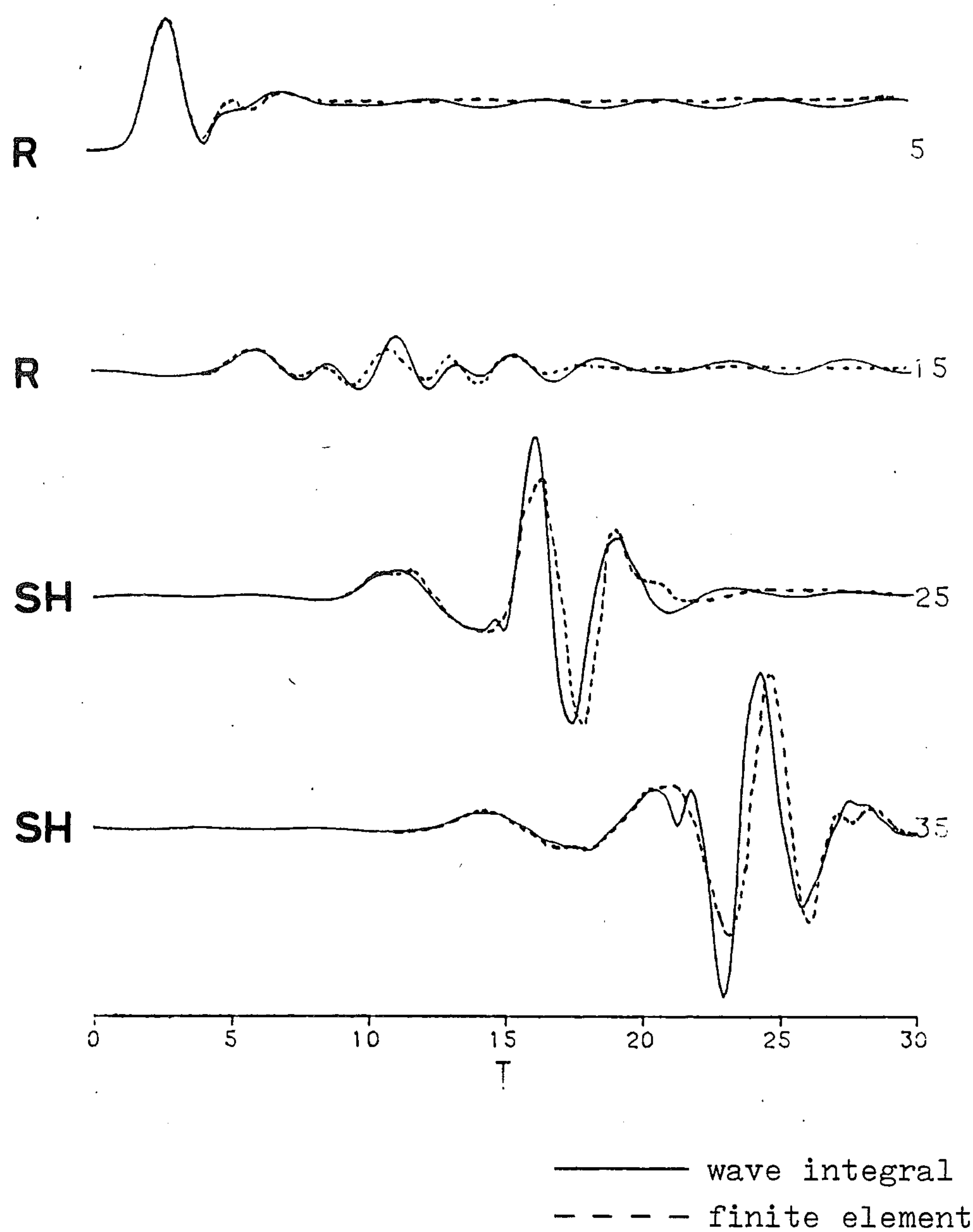


Figure 43. Comparison of wave integral solution with the finite element solution for the radial and tangential components due to a vertical strike-slip dislocation buried at a depth of 1 km in the two layers overlying half-space model of Table 3.

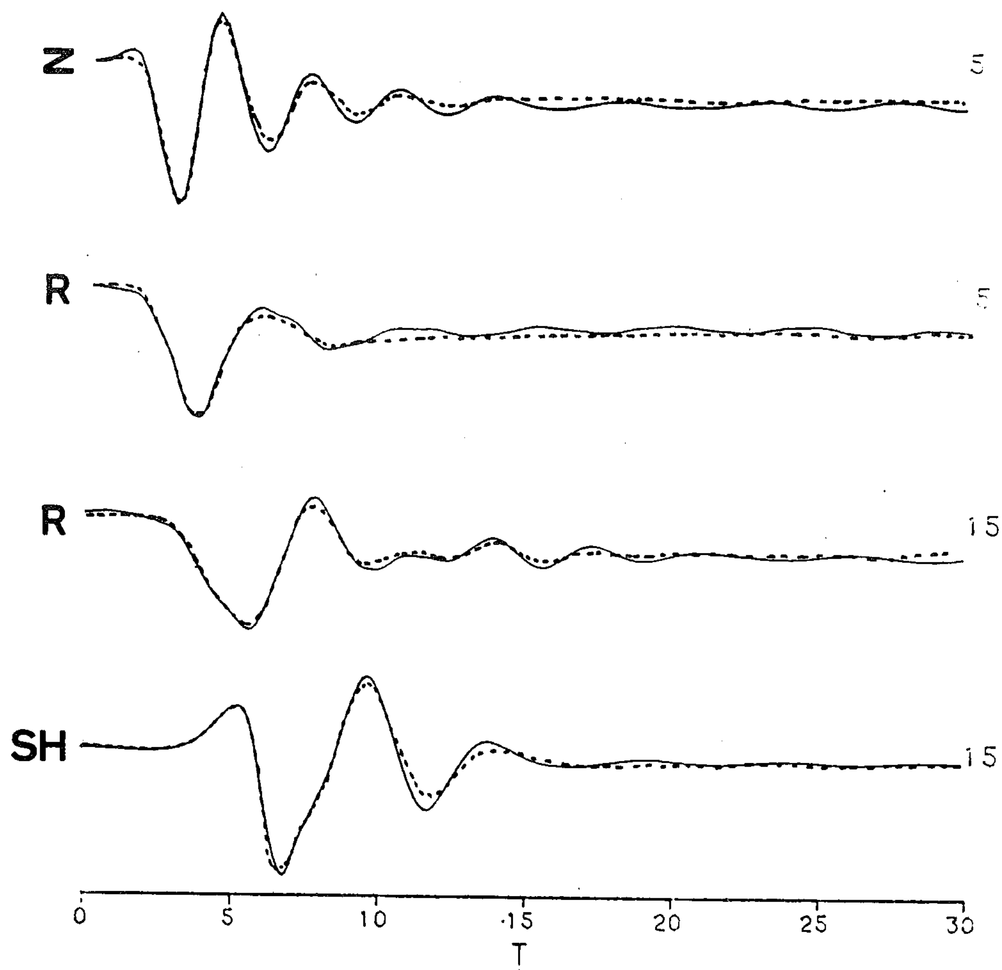


Figure 44. Results using the source and model of Figure 43, but for a source buried at the depth of 5 km.

Apsel (1979). The model consists of two layers overlying a half-space, as illustrated in Table 3. A vertical strike-slip dislocation source is considered and the source time function is a ramp of one second duration. The receivers with an azimuth of 22.5 degrees from the strike of the fault are located at the distances indicated at the ends of each traces. A low-pass filter with corner frequency Ø.5 Hz is applied to make the seismogram more visible. The results of the present method are computed up to 4 Hz before passing through the filter.

The comparsion results are shown in Figure 43 for a source at 1 km depth, and in Figure 44 for a source at 5 km. The components displayed are also indicated. The agreement in these figures is remarkable, especially in light of the vast differences between the two techniques. The slight deviation comes from the numerical errors of both methods and from slightly different low-pass filters used. However, the overall pattern still exhibits great consistency, especially for deeper sources, which is sufficient to confirm the success of the wave integral method.

Comparison to Modal Summation Method

In the previous examples we showed the comparisons with known complete solutions. The solutions include near-field as well as far-field terms, and the

TABLE 3

Two Layers Overlying Half-space Model

Thickness (km)	P vel (km/sec)	S vel (km/sec)	Density (g/cm^3)
2	3.0	1.73	1.67
2	5.0	2.887	2.89
–	6.0	3.46	3.46

integration is taken for different integral parts to create all kinds of signals. In the next example, several synthetic seismograms from Swanger and Boore (1978) and Heaton and Helmberger (1976,1977) are chosen to check against our pole contribution solutions, namely surface wave, as presented in chapter III. However, a solution from the locked mode approximation method is also included in an attempt to fit body wave signals.

Using the surface-wave modal superposition method of Harkrider (1964,1970), Swanger and Boore (1978) simulated ground displacement records from earthquakes in the Imperial Valley of California. Since the sediments in this valley form a prominent wave guide, surface waves are thought to be dominant. Heaton and Helmberger (1976,1977) generated several synthetic seismograms using the generalized ray method for this area. They have used enough rays in their summation so that their synthetics can be considered to be a near-complete solution. The effort of these studies was directed to find a suitable source model by time domain waveform fitting. However, the synthetic waveforms created provide us with a check of our eigenfunction programs. Figures 45 and 46 are the results showing the comparison of these different methods.

Using the El Centro model listed in Table 4, we

TABLE 4

Earth Models

Thickness (km)	P vel (km/sec)	S vel (km/sec)	Density (g/cm^3)
	El Centro Structure		
2.9	–	1.50	1.5
–	–	3.30	2.5
	Imperial Valley Structure		
0.95	2.0	0.88	1.8
1.15	2.6	1.50	2.35
3.8	4.2	2.40	2.6
–	6.4	3.70	2.8

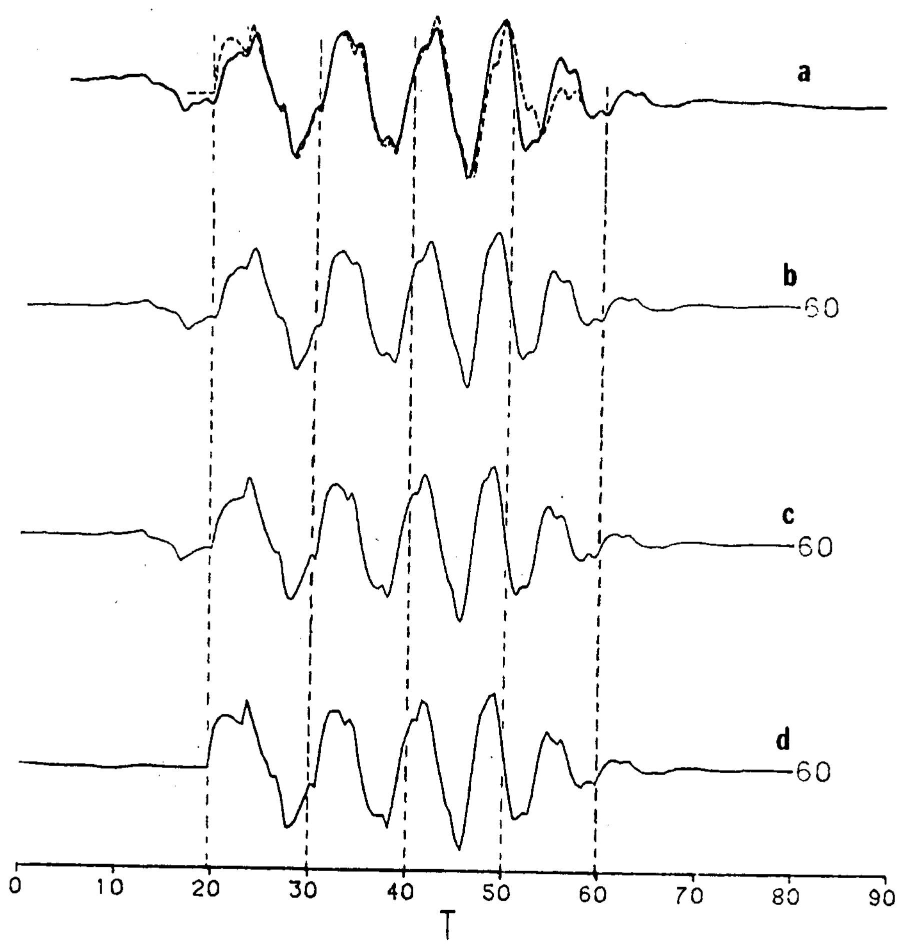


Figure 45. Comparison of Love wave synthetic ground displacements with Swanger and Boore's modal summation method (solid line in a) as well as Heaton and Helmberger's ray summation method (dashed line in a) of the 1968 Borrego Mountain earthquake. A vertical strike-slip source at 6 km depth and a symmetric triangular source time function of 1 second duration are used. The epicentral distance is 60 km and the azimuth is 8 degrees from the strike of the fault. The model used is the El Centro structure listed in Table 4. (b) is the result of eigenfunction programs but including only the first three modes which Swanger and Boore used. (c) is the result including all modes. (d) shows the synthetics from the locked mode approximation with a rigid layer at 200 km deep. Note that (d) successfully models the first arrival of ray theory solution.

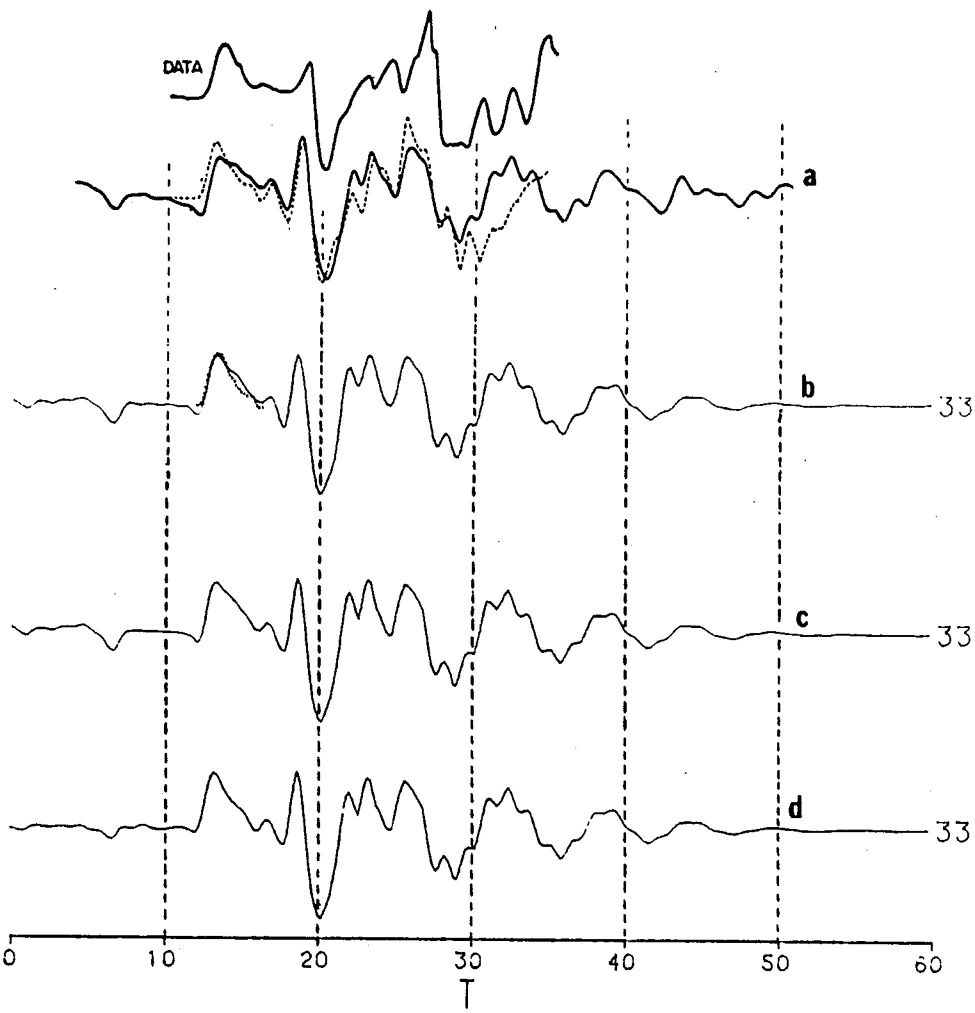


Figure 46. Same comparison as for Figure 45 but for the 1976 Brawley earthquake. The Imperial Valley structure (Table 4) of Heaton and Helmberger (1978) is used. The source is a vertical strike-slip point buried at 6.9 km, and the source time function is a 1.5 sec duration triangle. The top trace gives the real data (Heaton and Helmberger, 1978). Our solution (b), which is the summation of first five modes, shows a better fit of the first P arrival at 12 second than Swanger and Boore (1979). Again, the locked mode approximation (d) gives a good match to the ray theory solution in the front part of the record, but not in the rest.

generated surface wave seismograms after summing the first three modes ((b) in Figure 45) or all the modes ((c) in Figure 45) by eigenfunction programs. The match of these seismograms to Swanger and Boore's result (solid trace in 45a) is obvious. The agreement is superb and the phase coherence is nearly perfect. Furthermore, the seismogram from the locked mode approximation ((d) in Figure 45) fits the first P arrival of Heaton and Helmberger's solution (dashed trace in Figure 45a) quite well, although the later part of the trace seems not affected by the presence of a rigid cap layer.

Similar conclusions can be drawn for the comparison using the Imperial Valley model of Table 4. As shown in Figure 46, the result (b) of the present method seems to be a better fit comparison of the direct P arrival to ray theory solution than Swanger and Boore obtained. The reason might be a longer time window we have used to avoid the time aliasing. The locked mode approximation, again, shows a good match to the ray theory solution in the front part of the trace. These results serve to further validate the present method, despite the fact that various assumptions are inherent in the other methods. In addition, the comparison of (b) and (c) in Figures 45 or 46 also reveals that the number of modes Swanger and Boore used to make up their seismograms was sufficient.

CHAPTER VII

SUMMARY AND CONCLUSIONS

A detailed and complete derivation of wave integral theory for studying wave propagation in plane multi-layered media has been presented. The three-dimensional wave propagation problem is formulated and solved in the wavenumber-frequency domain. The layer responses at any wavenumber and frequency are given in terms of Haskell's layer matrices and the corresponding compound forms. The formalism presented was shown to be stable and accurate for computations. A contour integration is taken over horizontal wavenumber so as to automatically include all kinds of waves. A fast Fourier transform then gives the final time history.

The technique is not new. However, after a close step-by-step development, several insights were revealed. The most important was the finding of symmetry properties of the layer matrix and its compound counterparts. These properties enable us to build up the relation of the present method with eigenfunction theory and most of other currently used wave integral methods.

Eigenfunction solutions as expressed in analytic forms have the advantage of computational accuracy and efficiency. The special cases of high frequencies and complicated models can be handled without difficulty. New procedures were developed to find dispersion values. The eigenfunctions, energy integrals, and all other excited variables of surface waves could be evaluated with sufficient precision. Furthermore, the theoretical development also provided a rigorous basis for classical eigenfunction theory. This in turn opened a new, and potentially powerful method, which can be used to extract more information about the wave propagation through the earth.

The contour integration over the real branch cut is a difficult part of simulating body wave-like signals. The variation of the integrand along the real-k axis reflects the influence of leaky modes. After an intensive study of this effect, we were able to design a numerical integration procedure for this integration path. We also found that body waves and surface waves cannot be separated from each other. They are interconnected.

To test the versatility of the eigenfunction program developed, the locked mode approximation was introduced to simulate the signals from the branch cut contribution. Except for the computational speed, the test was highly successful. This method is worthy of further investigation.

One significant value of the present study is its ability to be extended to other methods. The reflection and reflectivity methods can be easily derived using the new system. As a result, the reflection and transmission properties of a layer boundary can be easily decomposed. With an arbitrary choice of reflection and/or transmission at interfaces, theoretical seismograms consisting of signals from a particular portion of the layered structure were obtained.

Source functions characterized by any first or second order point source were considered. A set of five fundamental forms was formed to represent all kinds of signals possibly excited in the earth. This expression, which isolates the direction-dependence in the final solution, is specially useful for focal mechanism studies.

A new way to find the instrument response, using least-squares inversion of calibration pulse waveforms, was developed. The method found the parameters which directly represent the instrument responses, even though the instrument-related constants are not given directly. These inversion procedures were performed in the time domain, and thus are easy to use. Finally, a digital z-transform technique, which was used to introduce the instrument effect in the time domain, was also discussed.

To verify the theory and numerical extension, a set of three validation studies was provided. The results of comparison to other known complete solutions serve to lend confidence in our method.

Because the models treated are plane stratified media, the present method can describe the wave propagation in the earth up to an epicentral distance, say, of 3000 km. Beyond this distance, the earth flattening correction from Biswas and Knopoff (1970) or Chapman (1973) should be considered. Furthermore, the dispersion curves for the models with an apparent low velocity zone sometimes become extremely complicated. Special care is needed to handle these particular cases.

An overall view of this wave theory system shows its completeness and ease of adaptation. With the advent of future computers, the method will become more and more important. It is expected that the present development and associated computer programs will prove increasingly useful in various areas of theoretical seismology and earthquake engineering.

APPENDIX A

Layer Matrix

The elements of layer matrix a are

$$a_{11} = \gamma \cosh \nu_\alpha z - (\gamma - 1) \cosh \nu_\beta z$$

$$a_{12} = k(\gamma - 1)\frac{\sinh \nu_\alpha z}{\nu_\alpha} - \frac{\gamma}{k}\nu_\beta \sinh \nu_\beta z$$

$$a_{13} = \frac{k}{\rho} \cosh \nu_\alpha z - \frac{k}{\rho} \cosh \nu_\beta z$$

$$a_{14} = -\frac{k^2}{\rho} \frac{\sinh \nu_\alpha z}{\nu_\alpha} + \frac{1}{\rho} \nu_\beta \sinh \nu_\beta z$$

$$a_{21} = -\frac{\gamma}{k}\nu_\alpha \sinh \nu_\alpha z + k (\gamma - 1) \frac{\sinh \nu_\beta z}{\nu_\beta}$$

$$a_{22} = -(\gamma - 1) \cosh \nu_\alpha z + \gamma \cosh \nu_\beta z$$

$$a_{23} = -\frac{\nu_\alpha}{\rho} \sinh \nu_\alpha z + \frac{k^2}{\rho} \frac{\sinh \nu_\beta z}{\nu_\beta}$$

$$a_{31} = -\rho \frac{\gamma(\gamma - 1)}{k} \cosh \nu_\alpha z + \rho\frac{\gamma(\gamma - 1)}{k} \cosh \nu_\beta z$$

$$a_{32} = -\rho (\gamma - 1)^2 \frac{\sinh \nu_\alpha z}{\nu_\alpha} + \rho \frac{\gamma^2}{k^2} \nu_\beta \sinh \nu_\beta z$$

$$a_{41} = -\rho \frac{\gamma^2}{k^2} \nu_\alpha \sinh \nu_\alpha z + \rho (\gamma - 1)^2 \frac{\sinh \nu_\beta z}{\nu_\beta}$$

$$a_{55} = \cosh \nu_\beta z \qquad a_{65} = \mu\nu_\beta \sinh \nu_\beta z$$

$$a_{56} = \frac{\sinh \nu_\beta z}{\mu\nu_\beta} .$$

The other elements are found by

$$a_{ij} = a_{5-j,5-i}$$ $a_{14} = a_{13}$

for i,j = 1,....,4, and

$$a_{ij} = a_{11-j,11-i}$$

for i,j = 5,6.

APPENDIX B

Compound Matrix

The compound matrix $\mathbf{E}^{-1}$ used in the main text is

$$E_N^{-1}|_{ij}^{12} = \frac{1}{4k\,\nu_{\alpha N}\,\nu_{\beta N}}\,[\,E^{-1}|_{12}^{12},\,E^{-1}|_{13}^{12},\,E^{-1}|_{14}^{12},\,E^{-1}|_{23}^{12},\,E^{-1}|_{24}^{12},\,E^{-1}|_{34}^{12}\,]_N$$

where

$$E^{-1}|_{12}^{12} = \rho^2\,[-\frac{\gamma^2}{k^2}\nu_\alpha\nu_\beta + (\gamma - 1)^2]$$

$$E^{-1}|_{13}^{12} = -\rho\nu_\alpha$$

$$E^{-1}|_{14}^{12} = \rho\,[\frac{\gamma}{k}\nu_\alpha\nu_\beta - (\gamma - 1)\,k\,]$$

$$E^{-1}|_{23}^{12} = E^{-1}|_{14}^{12}$$

$$E^{-1}|_{24}^{12} = -\rho\nu_\beta$$

$$E^{-1}|_{34}^{12} = \nu_\alpha\nu_\beta - k^2\ .$$

The compound matrix of layer matrix a is a 6x6 matrix whose components are

$$a\,|_{12}^{12} = CPCQ + 1 - a\,|_{14}^{14}$$

$$a\,|_{13}^{12} = (\,-CQX + k^2\,CPY\,)/\rho$$

$$a\,|_{14}^{12} = [(1 - CPCQ)\,(2\gamma - 1) + \frac{\gamma}{k^2}XZ + (\gamma - 1)\,k^2WY]\,k/\rho$$

$$\alpha \,\big|^{12}_{23} = \alpha \,\big|^{12}_{14}$$

$$\alpha \,\big|^{12}_{24} = (k^2\, CQW - CPZ)/\rho$$

$$\alpha \,\big|^{12}_{34} = [2(1 - CPCQ)\, k^2 + XZ + k^4\, WY]/\rho^2$$

$$\alpha \,\big|^{13}_{12} = \rho\, [-(\gamma - 1)^2\, CQW + \frac{\gamma^2}{k^2}\, CPZ]$$

$$\alpha \,\big|^{13}_{13} = CPCQ$$

$$\alpha \,\big|^{13}_{14} = (\gamma - 1)\, k\;\, CQW - \frac{\gamma}{k}\, CPZ$$

$$\alpha \,\big|^{13}_{23} = \alpha \,\big|^{13}_{14}$$

$$\alpha \,\big|^{13}_{24} = WZ$$

$$\alpha \,\big|^{14}_{12} = -\rho[(1-CPCQ)\, \frac{\gamma}{k}(\gamma-1)(2\gamma-1) + \frac{\gamma^3}{k^3}XZ + (\gamma-1)^3 k\;\, WY]$$

$$\alpha \,\big|^{14}_{13} = k\,(\gamma - 1)\, CPY - \frac{\gamma}{k}\, CQX$$

$$\alpha \,\big|^{14}_{14} = 1+2(1-CPCQ)\, \gamma(\gamma-1) + \frac{\gamma^2}{k^2}\, XZ + (\gamma-1)^2\, k^2\, WY$$

$$\alpha \,\big|^{14}_{23} = \alpha \,\big|^{14}_{14} - 1$$

$$\alpha \,\big|^{23}_{12} = \alpha \,\big|^{14}_{12}$$

$$\alpha \,\big|^{23}_{13} = \alpha \,\big|^{14}_{13}$$

$$\alpha \,\big|^{23}_{14} = \alpha \,\big|^{14}_{23}$$

$$\alpha \,\big|^{24}_{12} = \rho\; [\; CQX\; \frac{\gamma^2}{k^2} - CPY\; (\gamma - 1)^2]$$

$$\alpha \,\big|^{24}_{13} = XY$$

$$a\,\big|_{12}^{34} = \rho^2\,[2(1-CPCQ)\frac{\gamma^2}{k^2}(\gamma-1)^2 + \frac{\gamma^2}{k^4}\,XZ + (\gamma-1)^4\,WY]\ .$$

The other components can be found by

$$a\,\big|_{kl}^{ij} = a\,\big|_{5-j,5-i}^{5-l,5-k}$$

where

$$\gamma = \frac{2k^2}{k_\beta^2}$$

$$CPCQ = \cosh \nu_\alpha z \cdot \cosh \nu_\beta z$$

$$CPY = \cosh \nu_\alpha z \cdot Y$$

$$CPZ = \cosh \nu_\alpha z \cdot Z$$

$$CQW = \cosh \nu_\beta z \cdot W$$

$$CQX = \cosh \nu_\beta z \cdot X$$

$$XY = X \cdot Y$$

$$XZ = X \cdot Z$$

$$WY = W \cdot Y$$

$$WZ = W \cdot Z$$

with

$$W = \frac{\sinh \nu_\alpha z}{\nu_\alpha}$$

$$X = \nu_\alpha \cdot \sinh \nu_\alpha z$$

$$Y = \frac{\sinh \nu_\beta z}{\nu_\beta}$$

$$Z = \nu_\beta \cdot \sinh \nu_\beta z \quad .$$

APPENDIX C

Symmetry of Compound Matrix

Because of the particular symmetry existing in the layer matrix α, and those existing between α, E and their inverses respectively, we can reveal some interesting properties of a compound matrix obtained from these two kinds of matrices. In this appendix, these properties will be developed in somewhat rigorous mathematical terms for the compound matrices of order two which are used here. The equivalence of the third and fourth components of some of Haskell's compound matrices will consequently become apparent. Some of the properties derived in this appendix were used directly in the main text.

The special types of symmetry properties for the compound matrix can be summarized by two definitions:

(1) If two compound matrices A,B satisfy

$$A \,|^{ij}_{kl} = pB \,|^{\overline{kl}}_{ij} ,$$

where p is a constant coefficient

and $ij \leftrightarrow \overline{ij}$:

$$\begin{array}{lll} 12 \leftrightarrow 34 & 13 \leftrightarrow 24 & 14 \leftrightarrow 23 \\ 23 \leftrightarrow 14 & 24 \leftrightarrow 13 & 34 \leftrightarrow 12 , \end{array}$$

then A and B are called 'skew-symmetric'.

(2) If two compound matrices A,B satisfy

$$A\,|^{ij}_{kl} = pB\,|^{5-l,5-k}_{5-j,5-i}\ ,$$

then A and B are called '5-complemental'.

When B=A and p=1, such symmetric properties are called 'self skew-symmetric' or 'self 5-complemental'. When B = A^{-1} and p= $(-1)^{i+j+k+l}$ q, where q is a constant determined from A, such symmetric properties are called 'inverse skew-symmetric' or 'inverse 5-complemental'. 'Self skew-symmetry' is just a property indicating the symmetry of a compound matrix about its own skew diagonal axis if it is expanded in 6 by 6 form.

The compound matrix $\alpha\,|^{ij}_{kl}$ defined in Appendix B is found to be self skew-symmetric as well as self 5-complemental. The 5-complemental property actually comes from the skew-symmetry of the corresponding simple matrix α_{ij} , i.e., $\alpha_{ij} = \alpha_{5-j,5-i}$. Now we can see that the third and fourth components of $\alpha\,|^{ij}_{kl}$ must be equivalent in some manner. For example, the self skew-symmetry of $\alpha\,|^{12}_{14}$ is $\alpha\,|^{23}_{34}$ and its self 5-complement is $\alpha\,|^{14}_{34}$. Hence

$$\alpha\,|^{12}_{14} = \alpha\,|^{23}_{34} = \alpha\,|^{14}_{34} = \alpha\,|^{12}_{23}\ ,$$

where the last equality comes from the self skew-symmetry of $\alpha|^{14}_{34}$. Besides this, the compound matrix α is also inverse skew-symmetric as well as

inverse 5-complemental with coefficient $p=(-1)^{i+j+k+l}$.
Hence we have

$$a^{-1}|_{kl}^{ij} = (-1)^{i+j+k+l}\, a|_{ij}^{\bar{kl}} = (-1)^{i+j+k+l}\, a|_{5-j,5-i}^{5-l,5-k} = (-1)^{i+j+k+l} a|_{kl}^{ij} .$$

The compound matrix a has extremely good symmetry properties.

Similarly, we find that matrix E defined in equation (II-1-13) is inverse skew-symmetric with p = $(-1)^{i+j+k+l}$q and q = $4k^2\nu_\alpha\nu_\beta/\rho^2$. But it possesses neither the properties of self symmetry nor the property of inverse 5-complemental. With further investigation, this compound matrix is found to be inverse 5-complemental only for some of its components, which are ij=12, 34 kl=13, 24 for $E^{-1}|_{kl}^{ij}$, and ij=13, 24 kl=12, 34 for $E|_{kl}^{ij}$ with p equal to $(-1)^{i+j+k+l}$ times q or 1/q respectively. Other such properties occur for ij=14, 23 with p=-q for $E^{-1}|_{kl}^{ij}$ and kl=14, 23 with p=-1/q for $E|_{kl}^{ij}$. For example, $E^{-1}|_{kl}^{12}$ and $E|_{34}^{ij}$ are both inverse 5-complemental. Since these two are also inverse skew-symmetric, their third and fourth components are equivalent, i.e., $E^{-1}|_{14}^{12} = E^{-1}|_{23}^{12}$ $E|_{34}^{14} = E|_{34}^{23}$.

With the properties of compound a, E and E^{-1} revealed, it is not difficult to find that Haskell's matrices R and X which are composed of E^{-1} and a's, behave as E^{-1} , and Z which is composed of a's, behaves as a. In summary,

(1) $\mathbb{R}$ or $\mathbb{X}$ are inverse skew-symmetric.

(2) $R\,|^{12}_{kl}\, R\,|^{14}_{kl}\, R\,|^{23}_{kl}\, R\,|^{34}_{kl}\, R\,|^{ij}_{13}\, R\,|^{ij}_{24}$ and $R^{-1}|^{13}_{kl}\, R^{-1}|^{24}_{kl}\, R^{-1}|^{ij}_{12}\, R^{-1}|^{ij}_{14}$
$R^{-1}|^{ij}_{23}\, R^{-1}|^{ij}_{34}$ are inverse 5-complemental (*or* X).

(3) $\mathbb{Z}$ is self skew-symmetric and 5-complemental.

(4) $\mathbb{Z}$ is inverse skew-symmetric and 5-complemental.

APPENDIX D

Perturbation of Surface Wave Energy Integrals

In this appendix, we will evaluate $\frac{\partial}{\partial k} R\,|^{12}_{12}$ by using the variational principles. First, the eigenfunctions are perturbed by varying the wavenumber k about its stationary value. In such a perturbation state, the stresses at the free surface are small but not zero, and still satisfy

$$\mathbb{K}_N = \mathbb{R}\,\mathbb{B}_1$$

i.e.

$$\begin{bmatrix} 0 \\ 0 \\ A' \\ B' \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & R_{14} \\ R_{21} & R_{22} & R_{23} & R_{24} \\ R_{31} & R_{32} & R_{33} & R_{34} \\ R_{41} & R_{42} & R_{43} & R_{44} \end{bmatrix} \begin{bmatrix} \varepsilon \\ 1 \\ T_{z_1} \\ T_{r_1} \end{bmatrix},$$

where the vectors $\mathbb{K}_N$ and $\mathbb{B}_1$ have been normalized by U_{z_1}. The first two rows give

$$T_{z_1} = -[\,\varepsilon R\,|^{12}_{14} + R\,|^{12}_{24}\,] \,/\, R\,|^{12}_{34}$$

$$T_{r_1} = [\,\varepsilon R\,|^{12}_{13} + R\,|^{12}_{23}\,] \,/\, R\,|^{12}_{34} \qquad \text{(D-1)}$$

Note that ε is not the same ellipticity as defined in equation (III-1-4), since the system is no longer in an eigen state. If the system is really being characteristically excited, then $T_{z_1} = T_{r_1} = \emptyset$ and we have the same ellipticities as those in equation (III-1-4).

Return to the differential equation (II-1-11)

$$
\begin{aligned}
\frac{dT_z}{dz} &= -\rho U_z + k\ T_r \\
\frac{dT_r}{dz} &= (\ \rho - \xi\ \frac{k^2}{\omega^2}\ )\ U_r - k\sigma T_z \\
\frac{dU_z}{dz} &= -k\sigma U_r + \frac{\omega^2}{\lambda}\sigma T_z \\
\frac{dU_r}{dz} &= kU_z - \frac{\omega^2}{\mu}T_r\ ,
\end{aligned}
\tag{D-2}
$$

where

$$
\xi = \frac{4\mu\,(\lambda + \mu)}{\lambda + 2\mu}
$$

$$
\sigma = \frac{\lambda}{\lambda + 2\mu}\quad .
$$

Multiply the first equation of (D-2) by U_z and integrate it with respect to z from 0 to infinity:

$$
\begin{aligned}
U_z\ T_z\,|_0^\infty &= \int_0^\infty [\ -\rho U_z^2 + kU_z T_r + T_z \frac{d}{dz} U_z\ ]\ dz \\
&= \int_0^\infty [\ -\rho U_z^2 + kU_z T_r - k\sigma U_r T_z + \frac{\omega^2}{\lambda}\sigma T_z^2\ ]\ dz\ .
\end{aligned}
$$

Replacing the integrals by the summation over layers represented by I_{ij}'s,

$$
-U_{z_1} T_{z_1} = \sum_n -\rho_n (I_{22})_n + k\,(I_{24})_n - k\,\sigma_n (I_{13})_n + \omega^2\,\frac{\sigma_n}{\lambda_n}\,(I_{33})_n\ , \tag{D-3}
$$

where because of the conservation of energy, the radiation condition requires that U_{z_1} and T_{z_1} vanish at great depth. Similarly, the second equation in (D-2) can be integrated to become

$$
-U_{r_1} T_{r_1} = \sum_n (\rho_n - \xi_n \frac{k^2}{\omega^2})\,(I_{11})_n - k\,\sigma_n (I_{13})_n + k\,(I_{24})_n - \frac{\omega^2}{\mu_n}\,(I_{44})_n\ . \tag{D-4}
$$

Multiply equations (D-3) and (D-4) by ω^2 and mutually subtract,

$$-\omega^2 \left(U_{r_1}T_{r_1} - U_{z_1}T_{z_1} \right)$$
$$= \sum_n \left[\left(\rho_n - \xi_n \frac{k^2}{\omega^2}\right)(I_{11})_n + \rho_n (I_{22})_n - \omega^2 \frac{\sigma_n}{\lambda_n} (I_{33})_n - \frac{\omega^2}{\mu_n}(I_{44})_n \right] \omega^2 .$$

In this equation, the terms on the right are nothing but the Lagrangian of Rayleigh waves as given by equation (III-1-13),

$$-\omega^2 \left(U_{r_1}T_{r_1} - U_{z_1}T_{z_1}\right) = L_R \equiv \omega^2 I_0 - k^2 I_1 - 2kI_2 - I_3 \quad . \qquad \text{(D-5)}$$

Since all of the eigenfunctions are normalized (recall our definition of the energy integrals in equation III-1-10), $U_{z_1} = 1$ and $U_{r_1} = \varepsilon$. The terms inside the parentheses on the left side of equation (D-5) can be further expanded after substitution from equation (D-1),

$$\begin{aligned} & U_{r_1}T_{r_1} - U_{z_1}T_{z_1} \\ &= \varepsilon T_{r_1} - T_{z_1} \\ &= \left[\varepsilon\left(\varepsilon R\,|^{12}_{13} + R\,|^{12}_{23}\right) + \left(\varepsilon R\,|^{12}_{14} + R\,|^{12}_{24}\right)\right] / R\,|^{12}_{34} \\ &= \left[\varepsilon^2 R\,|^{12}_{13} R\,|^{12}_{13} + 2\varepsilon R\,|^{12}_{14} R\,|^{12}_{13} + R\,|^{12}_{24} R\,|^{12}_{13} \right] / \left(R\,|^{12}_{34} R\,|^{12}_{13}\right) , \end{aligned} \qquad \text{(D-6)}$$

where we have used

$$R\,|^{12}_{14} = R\,|^{12}_{23} \quad .$$

Since

$$R\,|^{12}_{12} R\,|^{12}_{34} + R\,|^{12}_{14} R\,|^{12}_{23} = R\,|^{12}_{13} R\,|^{12}_{24} \quad ,$$

equation (D-6) becomes

$$\varepsilon T_{r_1} - T_{z_1} = \frac{R\,|^{12}_{12}}{R\,|^{12}_{13}} + \frac{[\varepsilon R\,|^{12}_{13} + R\,|^{12}_{23}]^2}{R\,|^{12}_{34}\, R\,|^{12}_{13}} \quad .$$

Substitute into equation (D-5),

$$-\omega^2 \left[\frac{R\,|^{12}_{12}}{R\,|^{12}_{13}} + \frac{[\varepsilon R\,|^{12}_{13} + R\,|^{12}_{23}\,]^2}{R\,|^{12}_{34}\, R\,|^{12}_{13}} \right] = \omega^2 I_0 - k^2 I_1 - 2kI_2 - I_3 \quad .$$

Because the system slightly departs from the eigen state, it follows that a perturbation in k results in

$$-\omega^2 \frac{\frac{\partial}{\partial k} R\,|^{12}_{12}}{R\,|^{12}_{13}} = -2kI_1 - 2\,I_2 \quad ,$$

where we have set $R\,|^{12}_{12} \sim 0$ and consequently ε is very close to the value defined in equation (III-1-4), hence only the dominant term $\frac{\partial}{\partial k} R\,|^{12}_{12}$ is left. The group velocity as expressed in terms of integrals (Jeffreys, 1961) has the form

$$U = \frac{I_1 + I_2/k}{c\, I_0} \quad .$$

The result, i.e., the amplitude factor in terms of energy integrals, is obtained

$$\frac{R\,|^{12}_{13}}{\frac{\partial}{\partial k} R\,|^{12}_{12}} \frac{k}{\omega^2} = \frac{1}{2\,(\,I_1 + I_2/k\,)} = \frac{1}{2\,c\,U\,I_0} \quad . \qquad \text{(D-7)}$$

For the Love waves, the derivation is similar. Start with the differential equation

$$\frac{d}{dz} T_{\vartheta} = (\mu k^2 - \rho \omega^2) U_{\vartheta} .$$

Multiplying both sides by U_{ϑ} and integrating, yields

$$- T_{\vartheta_1} = -\omega^2 I_0 + k^2 I_1 + I_2 , \tag{D-8}$$

Then from

$$\begin{aligned} \mathbb{K}_N &= \mathbb{R}\, \mathbb{B}_1 \\ &= \mathbb{R} \; [\; 1 , \; T_{\vartheta_1}]^T , \end{aligned}$$

the perturbed stress T_{ϑ_1} is

$$T_{\vartheta_1} = - R_{55} / R_{56} .$$

Equation (D-8) becomes

$$\frac{R_{55}}{R_{56}} = - \omega^2 I_0 + k^2 I_1 + I_2 ,$$

Again, the right side is the Love wave Lagrangian. After taking the derivative with respect to k, and putting $R_{55} \sim 0$ we have

$$\frac{k \; R_{56}}{\frac{\partial}{\partial k} R_{55}} = \frac{1}{2 I_1} = \frac{1}{2 \, c \; U \; I_0} , \tag{D-9}$$

where the Love group velocity is just

$$U = \frac{I_1}{c \; I_0} \tag{D-10}$$

which is obtained by simply perturbing the Love wave Lagrangian.

APPENDIX E

Description of the Eigenfunction programs

To implement the eigenfunction theory as developed in chapter III, several FORTRAN computer programs have been set up using a DEC PDP 11/7Ø minicomputer in the department of Earth and Atmospheric Sciences, Saint Louis University. This appendix gives a brief description of these programs in order to facilitate their utilization and maintenance.

Because of the restriction of core size (about 6ØK bytes), we divided the computation into five separate programs which communicate via intermediate data files. The data files are stored in a high speed, large volume disc. The designation skeleton is inherited from Herrmann (1974, 1978b), although the theoretical and numerical techniques are modified. These programs are suited, especially, to short periods (less than 1 second) and complicated structures. Figure 47 shows this sequence of programs and their input/output. Table 5 lists the information to be used when running these programs.

(1) surface81: This program searches for dispersion values over the range of phase velocity between Ø.8 β_{min} and β_N for some fixed set of periods. It is suggested that these periods had better be arranged in

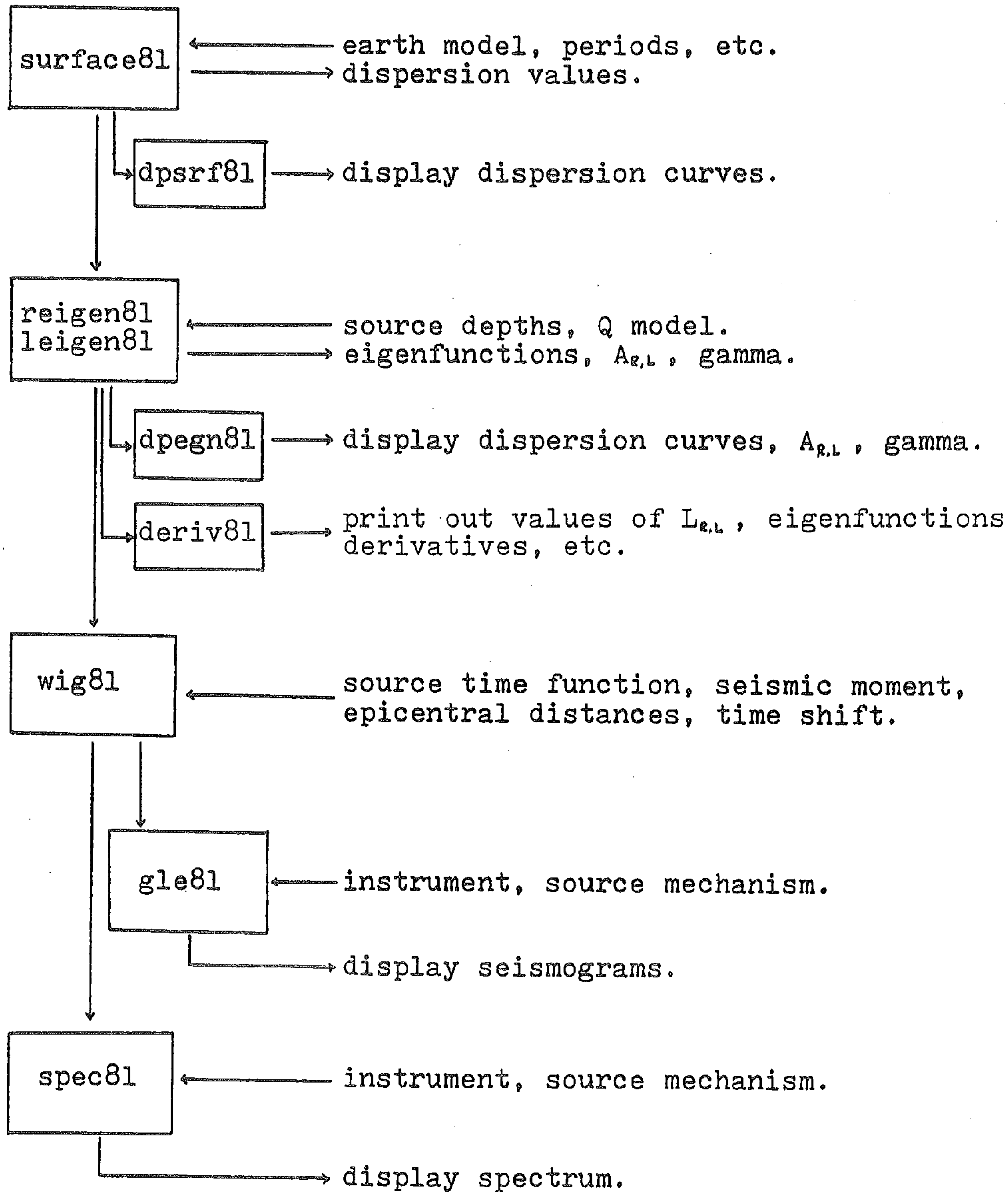


Figure 47. Flow chart of eigenfunction programs.

TABLE 5

Information for Synthesizing Seismograms

I. Sampling Information:

1. sampling rate, dt (sec): ____________

2. total sampling point: ____________

3. frequency range (Hz): ____________

II. Wave Type:

1. Rayleigh or Love? ____________

2. component (Z,R,T,NS,EW)? ____________

3. waveform (disp.,vel.,accel.)? ____________

III. Source Model:

1. source depth (Km): ____________

2. seismic moment (dyne-cm): ____________

3. source mechanism (dip,slip,strike): ____________

4. source time function: ____________

IV. Earth model:

1. velocity model:

2. Q model:

V. Receiver:

1. receiver position (r,Az): ____________

2. original time shift (t0): ____________

3. instrument (LP,SP,model): ____________

equally spaced frequency (inverse of period) so that a FFT can be used to transform spectral responses to time domain without using any sort of interpolation. The other input parameters are the structure which consists of the thickness of each layer, P-wave velocity, S-wave velocity and density. The number of layers and modes can be as large as 500.

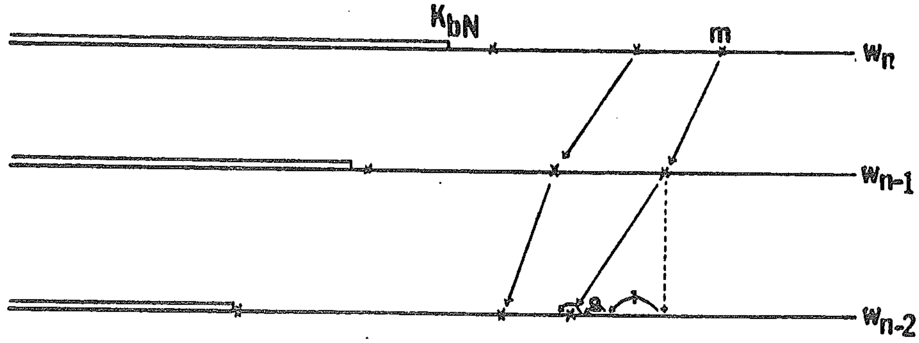
As pointed out in section 2.4, the search of poles is easier to carry out in the wavenumber-frequency plane than in the phase velocity-period plane. The reason is that equation (II-4-6) can give reasonably estimated steps to progress for a particular mode during the pole searching in the k-w plane. Figure 48 describes such a searching procedure. Note that the jump steps are frequency, wavenumber and mode dependence. In some cases, however, the dispersion values vary quite abruptly near the various P and S-wave layer velocities as illustrated in Figures 49 and 50. These figures display the dispersion curves of the CUS model at the high frequency range. The sharp bends and kinks result in very irregular pole spacing at a given frequency, which causes difficulty in pole searching. To overcome this, two more constraints to the values provided by equation (II-4-6) are needed:

$$\Delta k_{\min} = (k_{\max} - k_{\min}) / (200 + 200 \cdot frequency)$$

$$\Delta k_{\max} = 0.8 \cdot (k |_{mode=m} - k |_{mode=m-1}) |_{at\ previous\ freq}$$

These serve as the lower and higher bounds for the

JUMPING METHOD



First jump: $\frac{\Delta\omega}{c}$

Second jump: $k\frac{\Delta c}{c}$

which is restricted by

$$\Delta k_{\min} < k\frac{\Delta c}{c} < \Delta k_{\max}$$

$$\Delta k_{\min} = (\, k_{\max} - k_{\min}\,)/(\, 200 + 200 \cdot frequency \,)$$

$$\Delta k_{\max} = 0.8 * (\, k\,|_{mode=m} - k\,|_{mode=m-1}\,)\,|_{at\ previous\ freq}$$

Figure 48. Jumping method for searching poles.

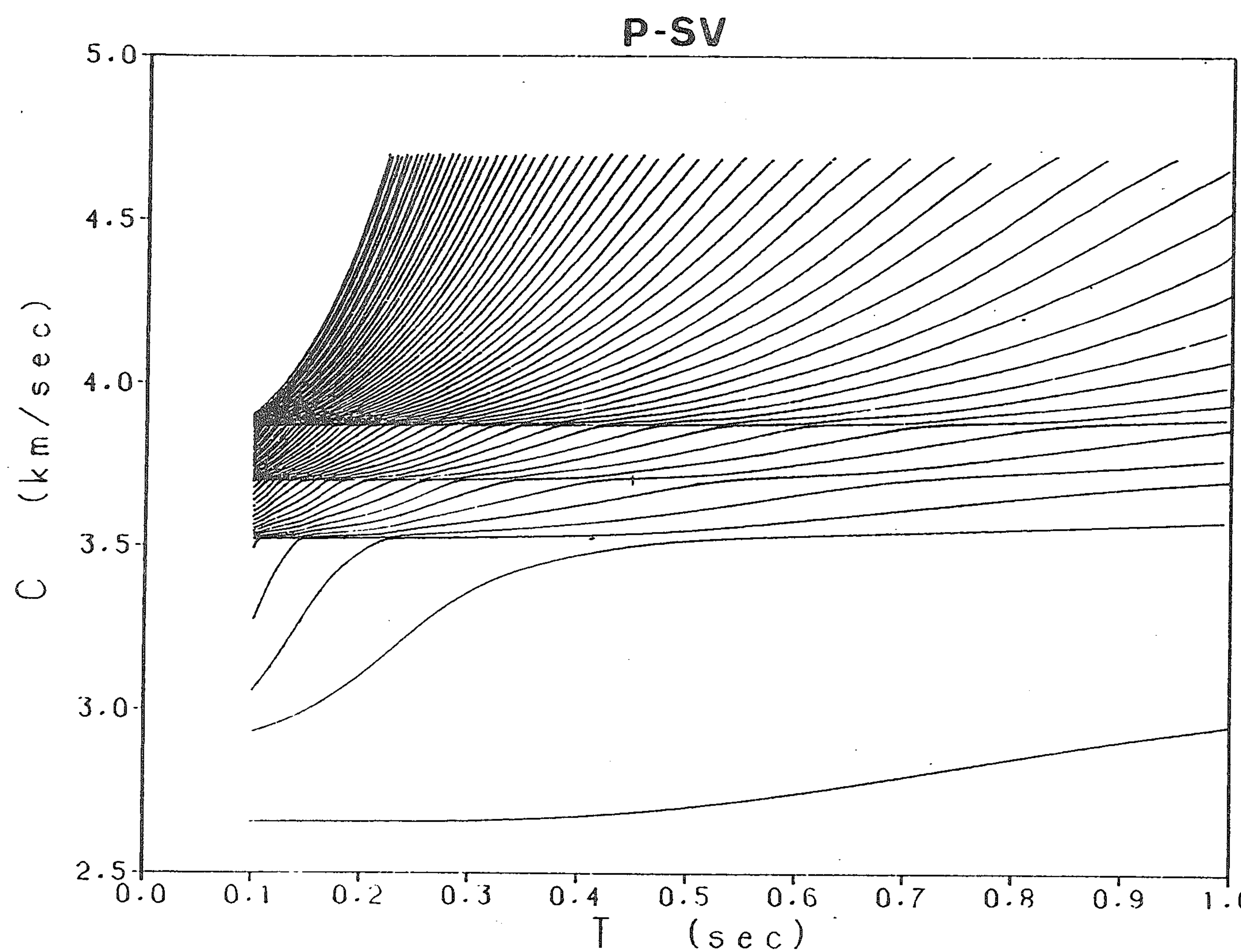


Figure 49. Phase velocity dispersion curves for the CUS model at short period range.

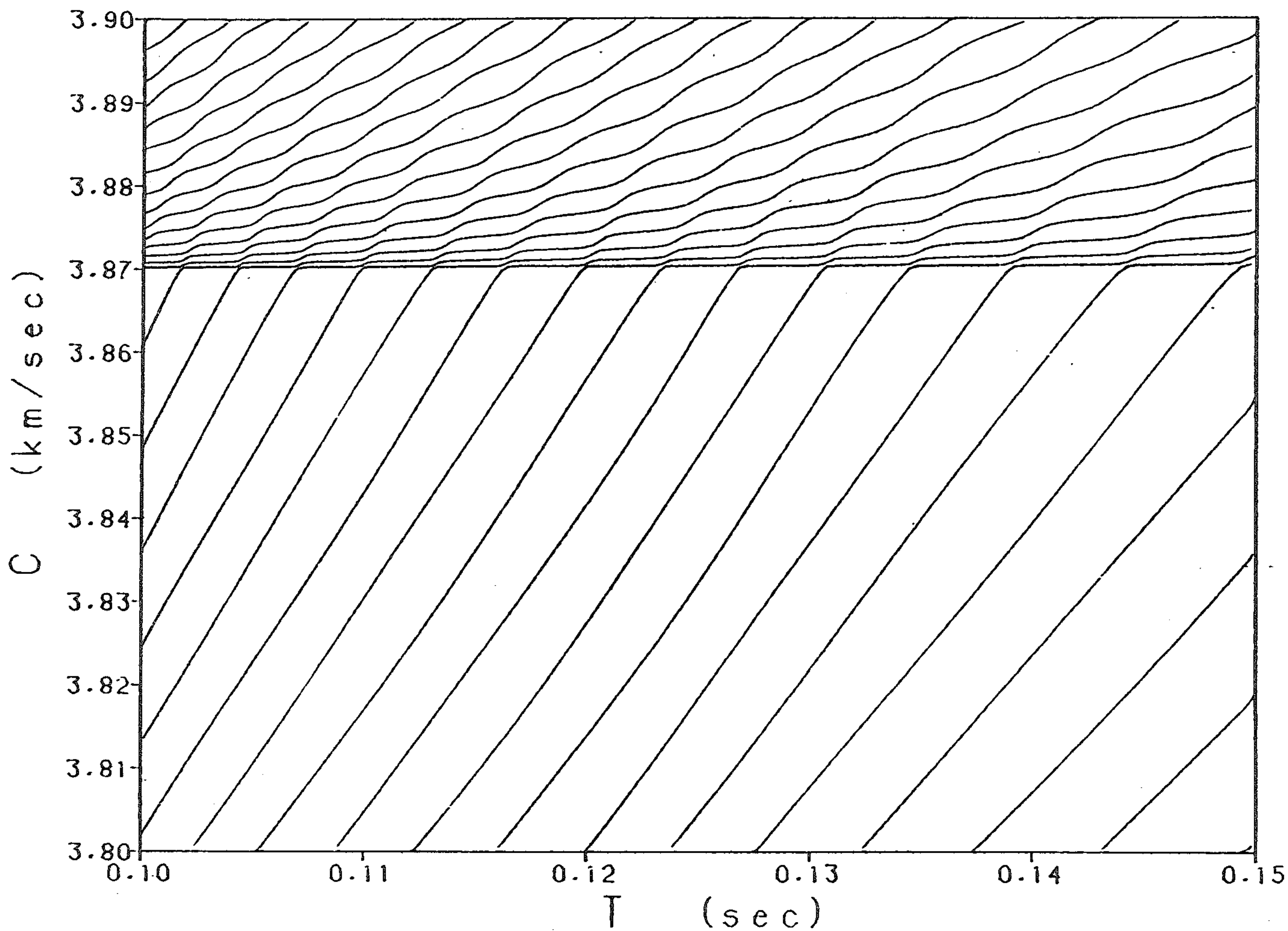


Figure 50. An expansion of Figure 49 for period between 0.10 and 0.15 second and phase velocity between 3.8 and 3.9 km/sec.

value of $\frac{k}{c}\Delta c$ from equation (II-4-6), i.e., if $\frac{k}{c}\Delta c$ is greater than Δk_{max} or smaller than Δk_{min}, Δk_{max} or Δk_{min} will be used to make several small jumps after the first jump $\frac{\Delta\omega}{c}$. This algorithm has been proved to be very efficient. For most models the pole is usually bracketed after not more than three or four searching jumps. The result can be displayed and checked by a plotting program 'dpsrf81'. All of the calculations in this program are performed in double precision.

(2) reigen81 (leigen81): This program computes the eigenfunctions and energy integrals which in turn give the group velocity, amplitude factor and attenuation coefficient. Reigen81 is for the P-SV case and leigen81 for SH. To save data file space, only the eigenfunction values at the source depth are stored. The source depth can be entered as several different values, and the output data for different source depths are stored in different output files. A Q model is input at this stage. The attenuation coefficients are determined using the perturbation theory of Anderson <u>et al</u> (1965). It is not difficult to incorporate any sort of frequency dependent Q by slightly modifying the subroutine 'gamma' in this program (Mitchell, 1980, 1981). The partial derivatives of phase velocity at different layer boundaries can also be stored for further study. To check the result, one can either print out the value of the Lagrangian for each dispersion pair or use the

program 'dpegn81' to plot group velocity, amplitude factor, etc. For good results, the curves plotted should be smooth, such as those shown in Figure 51. All of the calculations in these programs are done in double precision.

(3) wig81: This program computes the spectra for stations at a given set of epicentral distances. A particular source time function needs be specified here. In order to be able to treat long duration time series, a separate program 'bigfft' which executes FFT is involved. These two independent programs are connected by a 'system call'. Using this, the longest time series which can be created is 8192 points long, which is the largest dimension allowed in program 'bigfft'. In other programs only a vector with length of at most 1024 points is permitted. A linear interpolation is applied to generate enough data for spectra before taking the inverse Fourier transform, if necessary. The spectral data corresponding to different fundamental source types as listed in section 5.1 are generated and stored. The contributions from different modes can be separated as those used for higher mode study (Cheng and Mitchell, 1981).

(4) gle81 (spec81): The program 'gle81' reads the spectra data created in the previous program and generates the seismograms for each component at each

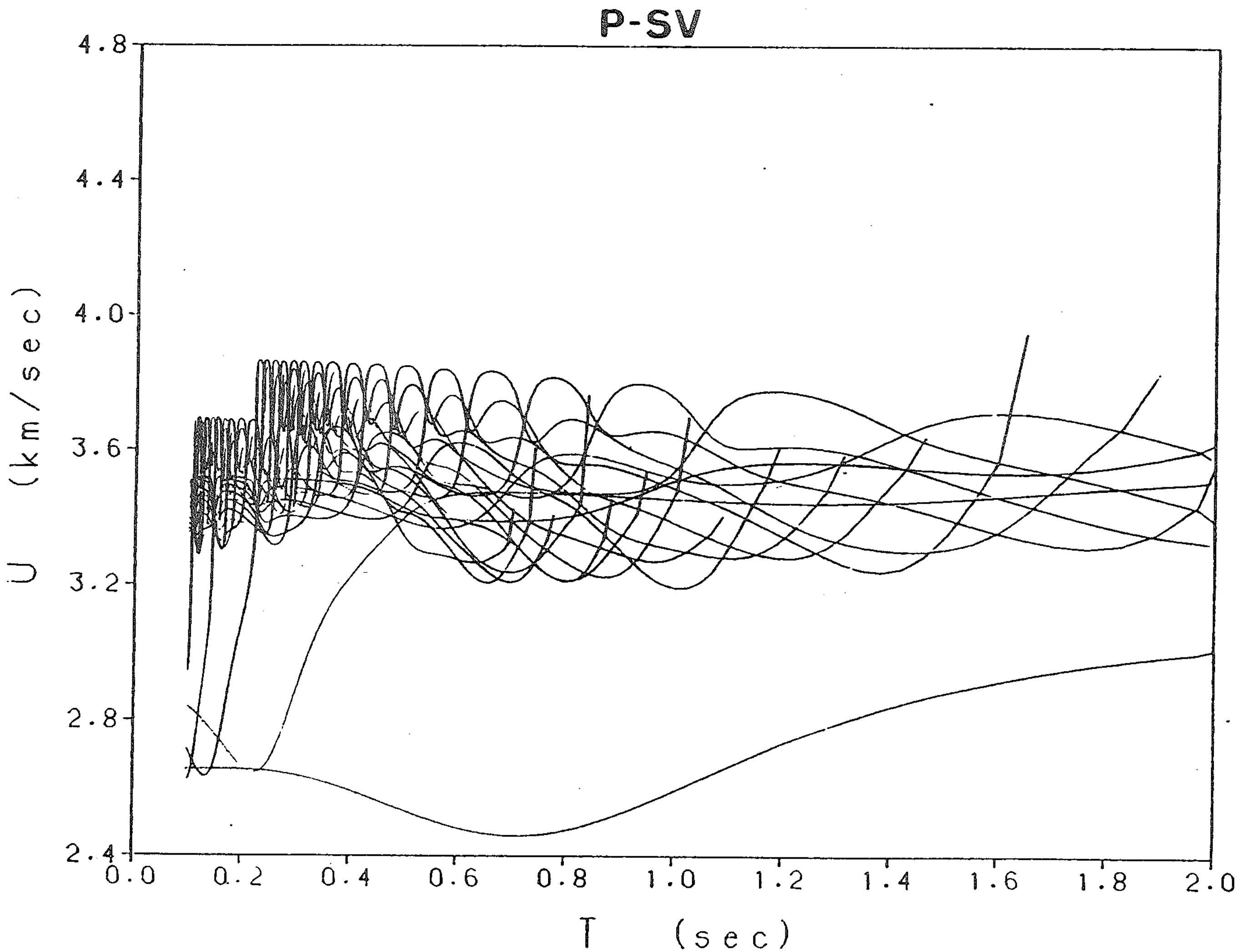


Figure 51. Group velocity dispersion curves for the CUS model. Note that the low order higher modes have group velocities around 3.5 km/sec.

receiver location after calling 'bigfft' to take inverse FFT. The seismogram is then displayed by a plotter. The program 'spec81' just reads and plots the spectra data. The focal mechanisms, as well as the receiver azimuths, are entered to take into account the directivity dependence. Several instrument responses, long and short periods, are included to take into account the instrument effect.

BIBLIOGRAPHY

Abo-Zena, A. (1979). Dispersion function computations for unlimited frequency values, Geophys. J. 58, 91-105.

Abramovici, F. (1968). Mode theory versus theoretical seismograms for a layered solid, Geophys. J. 16, 9-20.

Aki, K. and P. G. Richards (1980). Quantitative Seismology: Theory and Methods, Vol. I, W. H. Freeman and Company, San Francisco, 557pp.

Alsop, L. E. (1970). The leaky-mode period equation -a plane wave approach, Bull. Seism. Soc. Am. 60, 1989-1998.

Alterman, Z. S., H. Jarosch and C. L. Pekeris (1959). Oscillations of the earth, Proc. Roy. Soc. (London), A252, 80-95.

Anderson, D. L., A. Ben-Menahem, and C. B. Archambeau (1965). Attenuation of seismic energy in the upper mantle, J. Geophys. Res. 70, 1441-1448.

Apsel, R. J. (1979). Dynamic Green's Functions for Layered Media as Application to Boundary-value Problems, Ph.D. Dissertation, Univ. of Cal., San Diego, 349pp.

Baumgardt, D. R. (1980). Errors in matrix elements for the reflectivity method, J. Geophys. 48, 124-125.

Ben-Menahem, A. and D. G. Harkrider (1964). Radiation of seismic surface waves from buried dipolar point sources in a flat stratified earth, J. Geophys. Res. 69, 2605-2620.

Ben-Menahem, A. and S. J. Singh (1972). Computation of models of elastic dislocation in the earth, Methods in Computational Physics, Vol. 12, Academic Press, New York, 300-372.

Biswas, N. N. and L. Knopoff (1970). Exact earth-flattening calculation for Love waves, Bull. Seis. Soc. Am. 60, 1123-1137.

Bolt, B. A. and J. Dorman (1961). Phase and group velocities of Rayleigh waves in a spherical earth, J. Geophys. Res. 66, 2965-2981.

Bouchon, M. (1979). Discrete wave number representation of elastic wave fields in three-space dimensions, J. Geophys. Res. 84, 3609-3614.

Bouchon, M. (1981). A simple method to calculate Green's functions for elastic layered media, Bull. Seis. Soc. Am. 71, 959-971.

Bouchon, M. and K. Aki (1977). Discrete wave-number representation of seismic-wave fields, Bull. Seism. Soc. Am. 67, 259-277.

Brigham, E. O. (1974). The Fast Fourier Transform, Prentice-Hall, Englewood Cliffs, New Jersey, 252pp.

Burridge, R. and L. Knopoff (1964). Body force equivalents for seismic dislocations, Bull. Seism. Soc. Am. 54, 1875-1888.

Cagniard, L. (1962). Reflection and Refraction of Progressive Seismic Waves, trans. by E. A. Flinn and C. H. Dix, McGraw-Hill, New York.

Carpenter, E. W. (1966). A quantitative evaluation of teleseismic explosion records, Proc. Roy. Soc. (London), A290, 396-407.

Chapman, C. H. (1973). The earth flattening transformation in body wave theory, Geophys. J. 35, 55-70.

Chapman, C. H. (1978). A new method for computing synthetic seismograms, Geophys. J. 54, 481-518.

Cheng, C. C. and B. J. Mitchell (1981). Crustal Q structure in the United States from multi-mode surface waves, Bull. Seism. Soc. Am. 71, 161-181.

Claerbout, J. F. (1976). Fundamentals of Geophysical Data Processing: with Applications to Petroleum Prospecting, McGraw-Hill, New York, 274pp.

Cochran, M. D., A. F. Woeber, and J. Cl. DeBremaecker (1970). Body waves as normal and leaking modes 3. Pseudo modes and partial derivatives on (+,-) sheet, Rev. Geophys. 8, 321-357.

Dainty, A. M. (1971). Leaking modes in a crust with a surface layer, Bull. Seism. Soc. Am. 61, 93-107.

de-Hoop, A. T. (1960). A modification of Cagniard's method for pulse solving seismic pulse problem, Appl. Sci. Res. B8, 349-356.

Dunkin, J. W. (1965). Computation of modal solutions in layered, elastic media at high frequencies, Bull. Seism. Soc. Am. 55, 335-358.

Embree, P., J. P. Burg, and M. M. Backus (1963). Wide-band velocity filtering- the pie-slice process, Geophys. 28, 948-974.

Ewing, W. M., W. S. Jardetzsky, and F. Press (1957). Elastic Waves in Layered Media, McGraw-Hill, New York, 380pp.

Frasier, C. W. (1970). Discrete time solution of plane P-SV waves in a plane layered medium, Geophys. 35, 197-219.

Fuchs, K. (1968). Das reflexions-und Transmission vermogen eines geschichteten Mediums mit beliebiger Tiefen-Verteilung der elastichen Modulen und der Dichte fur schragen Einfall ebener Wellen. Z. Geophys. 34, 389-413.

Fuchs, K. (1971). The method of stationary phase applied to the reflection of spherical waves from transition zones with arbitrary depth dependent elastic moduli and density, J. Geophys. 37, 89-117.

Fuchs, K. and G. Müller (1971). Computation of synthetic seismograms with the reflectivity method and comparison with observations, Geophys. J. 23, 417-433.

Gilbert, F. (1964). Propagation of transient leaking modes in a stratified elastic waveguide, Rev. Geophys. 2, 123-154.

Gilbert, F. and G. E. Backus (1966). Propagator matrices in elastic wave and vibration problems, Geophys. 31, 326-332.

Harkrider, D. G. (1964). Surface waves in multi-layered elastic media. I. Rayleigh and Love waves from buried sources in multi-layered elastic half-space. Bull. Seism. Soc. Am. 54, 627-679.

Harkrider, D. G. (1970). Surface waves in multi-layered elastic media. II. Higher mode spectra and spectral ratios from point sources in plane layered earth models, Bull. Seism. Soc. Am. 60, 1937-1987.

Harkrider, D. G. (1976). Potentials and displacements for two theoretical seismic sources, Geophys. J. 47, 97-133.

Harkrider, D. G. and D. L. Anderson (1966). Surface wave energy from point sources in plane layered Earth models, J. Geophys. Res. 71, 2967-2980.

Harvey, D. J. (1981). Seismogram synthesis using normal mode superposition: the locked mode approximation, Geophys. J. 66, 37-69.

Haskell, N. A. (1953). The dispersion of surface waves in multilayered media, Bull. Seism. Soc. Am. 43, 17-34.

Haskell, N. A. (1963). Radiation pattern of Rayleigh waves from a fault of arbitrary dip and direction of motion in a homogeneous medium, Bull. Seism. Soc. Am. 53, 619-642.

Haskell, N. A. (1964). Radiation pattern of surface waves from point sources in a multi-layered medium, Bull. Seism. Soc. Am. 54, 377-393.

Haskell, N. A. (1966). The leakage attenuation of continental crustal P waves, J. Geophys. Res. 71, 3955-3967.

Heaton, T. H. and D. V. Helmberger (1976). Predictability of strong ground motion in the Imperial Valley: modeling the M4.9, November 4, 1976 Brawley earthquake, Bull. Seism. Soc. Am. 66, 31-48.

Heaton, T. H. and D. V. Helmberger (1977). A study of the strong ground motion of the Borrego Mountain, California, earthquake, Bull. Seism. Soc. Am. 67, 315-330.

Helmberger, D. V. (1968). The crust-mantle transition in the Bering Sea, Bull. seism. Soc. Am. 58, 179-214.

Herrmann, R. B. (1974). Surface Wave Generation by Central United States Earthquakes, Ph.D. Dissertation, Saint Louis University, 263 pp.

Herrmann, R. B. (1978a). A note on causality problems in the numerical solution of elastic wave propagation in cylindrical coordinate systems, Bull. Seism. Soc. Am. 68, 117-124.

Herrmann, R. B. (1978b, Editor). Computer Programs in Earthquake Seismology, Vol. 2: Surface Wave Programs, Department of Earth and Atmospheric Sciences, Saint Louis University.

Herrmann, R. B. (1979). SH-wave generation by dislocation sources -a numerical study, Bull. Seism. Soc. Am. 69, 1-15.

Herrmann, R. B. and C. Y. Wang (1979). SH-A Computer Program for Generating Far-field Tangential Time Histories for Point Earthquake Sources, Department of Earth and Atmospheric Sciences, Saint Louis University.

Hildebrand, F. B. (1965). Methods of Applied Mathematics, Prentice-Hall, Englewood Cliffs, New Jersey, 362pp.

Hron, F. (1972). Numerical methods of ray generation in multilayered media, Methods in Computational Physics , Vol. 11, Academic Press, New York, 1-34.

Hudson, J. A. (1969a). A quantitative evaluation of seismic signals at teleseismic distances -I. Radiation from point sources, Geophys. J. 18, 233-249.

Hudson, J. A. (1969b). A quantitative evaluation of seismic signals at teleseismic distances -II. Body waves and surface waves from an extended source, Geophys. J. 18, 353-370.

Jarosch, H. and A. R. Curtis (1973). A note on the calibration of the electromagnetic seismograph, Bull. Seism. Soc. Am. 63, 1145-1155.

Jeffreys, H. (1961). Small correction in the theory of surface waves, Geophys. J. 6, 115-117.

Johnson, L. R. (1974). Green's function for Lamb's problem, Geophys. J. 37, 99-131.

Keilis-Borok, V. I. (1960). Interference surface waves, M., Izd-vo Acad. Sci. USSR, trans. by R. Herrmann.

Kennett, B. L. N. (1974). Reflections, rays and reverberations, Bull. Seism. Soc. Am. 64, 1685-1696.

Kennett,B. L. N. (1975). Theoretical seismogram calculation for laterally varying crust structures, Geophys. J. 42, 579-589.

Kennett, B. L. N. (1980). Seismic waves in a stratified halfspace. II. Theoretical seismograms, Geophys. J. 61, 1-10.

Kennett, B. L. N., N. J. Kerry, and J. H. Woodhouse (1978). Symmetries in the reflection and transmission of elastic waves, Geophys. J. 52, 215-230.

Kennett, B. L. N. and N. J. Kerry (1979). Seismic waves in a stratified half space, Geophys. J. 57, 557-583.

Kerry, N. J. (1981). Synthesis of seismic surface waves, Geophys. J. 64, 425-446.

Kind, R. (1976). Computation of reflection coefficients for layered media, J. Geophys. 42, 191-200.

Kind, R. (1978). The reflectivity method for a buried source, J. Geophys. 44, 603-612.

Kind, R. (1979). Extensions of the reflectivity method, J. Geophys. 45, 373-380.

Kind, R. and G. Müller (1975). Computation of SV waves in realistic earth models, J. Geophys. 41, 149-172.

Kind, R. and G. Müller (1977). The structure of the outer core from SKS amplitudes and travel times, Bull. Seism. Soc. Am. 67, 1541-1554.

Knopoff, L. (1964). A matrix method for elastic wave problems, Bull. Seism. Soc. Am. 54, 431-438.

Kulhanek, U. (1979). Introduction to Digital Filtering in Geophysics, Elsevier Scientific Publishing Company, New York, 168pp.

Lamb, H. (1904). On the propagation of tremors over the surface of an elastic sold, Proc. Roy. Soc. (London), A203, 1-42.

Lapwood, E. R. (1949). The disturbance due to a line source in a semi-infinite elastic medium, Proc. Roy. Soc. (London), A243, 63-100.

Laster, S. J., J. F. Foreman, and A. F. Linville (1965). Theoretical investigation of modal seismograms for a layer over half-space, Geophys. 30, 571-596.

Levshin, A. L. and A. Z. Yanson (1971). Surface waves in vertically and radially inhomogeneous media (English translation), Computer Programs in earthquake Seismology, Vol. 2: Surface Wave Programs, Department of Earth and Atmospheric Sciences, Saint Louis University, A1-A35.

Love, A. E. H. (1944). A Treatise on the Mathematical Theory of Elasticity, Dover Publications, New York, 643pp.

Luh, P. C. (1977). A scheme for expressing instrumental responses parametrically, Bull. Seism. Soc. Am. 67, 957-969.

Menke, W. (1979). Comment on 'Dispersion function computation for unlimited frequency values' by Anas Abo-Zena, Geophys. J., 59, 315-323.

Mitchell, B. J. and M. Landisman (1969). Electromagnetic seismograph constants by least-squares inversion, Bull. Seism. Soc. Am. 59, 1335-1348.

Mitchell, B. J. (1980). Frequency dependence of shear wave internal friction in the continental crust of eastern North America, J. Geophys. Res. 85, 5212-5218.

Mitchell, B. J. (1981). Regional variation and frequency dependence of Qb in the crust of the United States, Bull. Seism. Soc. Am. 71, 1531-1538.

Mitronovas, W. (1976). Accuracy in phase determination of LP electromagnetic seismograph based on transient calibration pulse, Bull. Seism. Soc. Am. 66, 97-104.

Müller, G. (1971). Approximate treatment of elastic body waves in media with spherical symmetry, Geophys. J., 23, 431-449.

Müller, G. and R. Kind (1976). Observed and computed seismograms for the whole earth, Geophys. J. 44, 699-716.

Nuttli, O. W., W. Stauder and C. Kisslinger (1969). Travel time tables for earthquakes in the central United States, Earthquake Notes 40, 19-28.

Oliver, J. and M. Major (1960). Leaking modes and the PL phase, Bull. Seism. Soc. Am. 50, 165-180.

Oppenheim, A. V. and R. W. Schafer (1975). Digital Signal Processing, Prentice-Hall, Englewood Cliffs, New Jersey, 585pp.

Pao, Y. H. and R. R. Gajewski (1977). The generalized ray-theory and transient response of layered elastic solids, Physical Acoustics, Vol. 13, Academic Press, New York.

Papoulis, A. (1962). The Fourier Integral and Its Application, McGraw-Hill, New York, 318pp.

Phinney, R. A. (1961). Leaking modes in the crustal waveguide. Part 1. The Oceanic PL wave, J. Geophys. Res. 66, 1445-1470.

Pilant, W. L. (1979). Elastic Waves in the Earth, Elsevier Scientific Publishing Company, New York, 493pp.

Press, F., D. Harkrider, and C. A. Seafeldt (1961). A fast convenient program for computation of surface-wave dispersion curves in multilayered media, Bull. Seism. Soc. Am. 51, 495-502.

Randall, M. J. (1967). Fast programs for layered half-space problems, Bull. Seism. Soc. Am. 57, 1299-1315.

Robinson, E. A. and S. Treitel (1980). Geophysical Signal Processing, Prentice-Hall, Englewood Cliffs, New Jersey, 467pp.

Rosenbaum, J. H. (1960). The long-time response of a layered elastic medium to explosive sound, J. Geophys. Res. 65, 1577-1613.

Saito, M. (1967). Excitation of free oscillations and surface waves by a point source in a vertically heterogeneous earth, J. Geophys. Res. 72, 3689-3699.

Schwab, F. and L. Knopoff (1970). Surface-wave dispersion computations, Bull. Seism. Soc. Am. 60, 321-344.

Seidl, D. (1980). The simulation problem of broad-band seismograms, J. Geophys. 48, 84-93.

Su, S.S. and J. Dorman (1965). The use of leaking modes in seismogram interpretation and in studies of crust-mantle structure, Bull. Seism. Soc. Am. 55, 989-1021.

Swanger, H. J. and D. M. Boore (1978). Simulation of strong-motion displacements using surface-wave modal superposition, Bull. Seism. Soc. Am. 68, 907-922.

Takeuchi, H. and M. Saito (1972). Seismic surface waves, Methods in Computational Physics, Vol. 12, Academic Press, New York, 217-295.

Thomson, W. T. (1950). Transmission of elastic waves through a stratified solid medium, J. Appl. Phys. 21, 89-93.

Thrower, E. N. (1965). The computation of the dispersion of elastic waves in layered media, J. Sound Vib. 2, 210-226.

Tsai , Y. B. and K. Aki (1970). Precise focal depth determination from amplitude spectra of surface waves, J. Geophys. Res. 75, 5729-5743.

Vered, M. and A. Ben-Menahem (1976). Generalized multipolar ray theory for surface and shallow sources, Geophys. J. 45, 185-198.

Vlaar, N. J. (1966). The field from an SH-point source in a continuously layered inhomogeneous medium, 1. The field in a layer of finite depth, Bull. Seism. Soc. Am. 56, 715-724.

Wang,C. Y. and R. B. Herrmann (1980). A numerical study of P-, SV-, and SH-wave generation in a plane layered medium, Bull. Seism. Soc. Am. 70, 1015-1036.

Watson, T. H. (1970). A note on fast computation of Rayleigh wave dispersion in layered elastic halfspace, Bull. Seism. Soc. Am. 60, 161-166.

Watson, T. H. (1972). A real frequency, complex wavenumber analysis of leaking modes, Bull. Seism. Soc. Am. 62, 369-384.

Wiggins, R. A. and J. A. Madrid (1974). Body wave amplitude calculation, Geophys. J. 37, 423-433.

VITA AUCTORIS

Chien-Ying Wang was born on May 24, 1951 in Tainan, Taiwan, Republic of China. He attended the National Central University, Chung-Li, Taiwan, from 1969 to 1973, and received his B.S. degree in Physics in 1973. After two years of military duty, he then became a graduate student at the Oceanography Institute of National Taiwan University, Taiwan, and received an M.S. degree in Geophysics in 1977. During the years 1977 to 1981 he has been a graduate student and research assistant in the Department of Earth and Atmospheric Sciences of Saint Louis University, St. Louis, Missouri.

COMPUTER PROGRAMS

The computer programs listed here represent substantial improvements to those given by Herrmann (1978b). All programs are written in FORTRAN 77 and are run on a PDP 11/70 under the UNIX* operating system. This system makes use of the separate Instruction and Data space feature of the PDP 11/70 processor. Thus even though the PDP 11/70 is a 16 bit minicomputer, the text and data spaces can each be as large as 64 K bytes. However, because of the size limitations of even this machine, one program of Herrmann (1978b), wiggle, was split into two programs, wig81 and gle81. Double precision is used, which means that a floating point number uses 64 bits, or has about 16 significant figures.

The only non-standard usage is the

```
call system ('bigfft', kturn)
```

Under UNIX, this permits one program to initiate another program and waits until that program completes before proceeding. The program bigfft performs a Fast Fourier Transform of length 2n by performing a Fast Fourier Transform of length n using the constraint that the time series is purely real. The analog of "call system" may not exist on other computers, but then those computers may be more than 16 bit machines and hence the subterfuge used here may be removed by replacing the sequence of writing, reading and closing temporary files with array access and the "call system" by a

*UNIX is a trademark of Bell Laboratories.

"call bigfft" where "bigfft" is now a subroutine rather than a separate program.

The plotting programs make use of subroutine calls to a CALCOMP* plotter. These are

```
CALL PLOTS (0, 0, LDEV)                    open plotter
CALL PLOT (X, Y, ±IPEN)                    move plotter to X,Y
                                           with pen up (IPEN=
                                           ±3) or pen down
                                           (IPEN=±2) and reset
                                           origin at new position
                                           if IPEN=negative
CALL FACTOR (FACT)                         multiply all
                                           movements by FACT
CALL NEWPEN (INP)                          use pen INP
CALL SYMBOL (X, Y, HEIGHT, ICHAR, ANGLE, + NCHAR)   write
CALL SYMBOL (X, Y, HEIGHT, INTEQ, ANGLE, -ICODE)  text array
                                           plot symbol INTEQ with
                                           pen up/down, ICODE=-1/-1
                                           during move
CALL NUMBER (X, Y, HEIGHT, FLOATNUMB, ANGLE, ±NDEC)
                                           write number at X, Y
CALL SCALE (ARRAY, AXLEN, NPTS, ±INC) define
                                           FIRSTV ARRAY (NPTS+1)
                                           DELTAV ARRAY (NPTS +
                                           INC + INC +1)
CALL AXIS (X, Y, TITLE, ±NCHAR, AXLEN, ANGLE, FIRSTV, DELTAV)
                                           plot axis with label
CALL LINE (XARRAY, YARRAY, NPTS, INC, LINTYP, INTEQ)
                                           draw line with/without
                                           symbol at each point
                                           FIRSTV and DELTAV
                                           must be defined for each
                                           array
```

*CALCOMP is a trademark.

Eigenfunction Programs

```
c      program surface81.f
c
c      f77 -i -I2 surface81.f -o surface81
c
c      This program calculates the dispersion values for any
c      layered model, any frequency, and any mode.
c
c      This program will accept one liquid layer at the surface.
c      In such case ellipticity of rayleigh wave is that at the
c      top of solid array.  Love wave communications ignore
c      liquid layer.
c
c      Program developed by Robert B Herrmann Saint Louis
c      univ. Nov 1971, and revised by C. Y. Wang on Oct 1981.
c
c      All processes are performed in the wavenumber-frequency
c      domain.
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c      INPUT:
c
c      1. mmax,mode     (eg. 2,1000)
c         -mmax: >0 number of layers including halfspace
c                <0 end program
c                =0 use previous model with new options
c         -mode:    number of modes for which dispersion curves
c                   are desired. (1000 is the maximum)
c
c      2. d(i),a(i),b(i),rho(i)    (eg.   40.0,6.15,3.55,2.8
c                     i=1,mmax                8.09,4.67,3.3    )
c         -d(i): layer thickness (km)
c         -a(i): P wave velocity (km/sec)
c         -b(i): S wave velocity (km/sec)
c         -rho(i): density          (gm/cm3)
c
c      3  igphl,igphr,icut,idispl,idispr,ipunch
c                          (eg.  1,1,1,1,1,1,3.0)
c         -igphl  =1 Love wave period equation plot
c                 =0 not plot
c         -igphr  =1 Rayleigh wave period equation plot
c                 =0 not plot
c         -icut   =1 find the cutoff periods of Love and
c                    Rayleigh waves.
c                 =0 not
c         -idispl =1 find the dispersion values and amplitude
c                    factor for Love wave
c                 =0 not
c         -idispr =1 find the dispersion values, amplitude
c                    factor and ellipticity for Rayleigh wave
c                 =0 not
c         -ipunch =1 store and print the output of dispersion
c                    values.
```

```
c                  =0 store the dispersion values only
c                 =-1 print the dispersion values only
c
c     4. if igphl or igphr =1
c        kk,c1,c2,dc
c        t(i) i=1,kk          (eg.  10,3.5,8.1,0.1
c                                   1,2,3,4,5,6,7,8,9,10     )
c        -kk: number of periods along abscissa
c        -c1: lower phase velocity limit
c        -c2: upper phase velocity limit
c        -dc: phase velocity increment
c
c        -t(i): periods for which period equation is computted
c
c     5. if icut =1
c        kmax,t1,dt,c1        (eg. 4,18.0,-0.2,4.67)
c        -kmax: number of cutoff periods to be found for c1
c        -t1:   initial period of search
c        -dt:   period increment (can be negative)
c        -c1:   phase velocity for which kmax cutoffs are desired
c
c     6. if idispl or idispr =1
c        kmax,t1,dt,c1
c        if t1=0.0
c        t(i) i=1,kmax    (eg. 10,0.0,0.0,3.5,0.005
c                 1,1.25,1.5,1.75,2.0,2.25,2 5,2.75,3.0,3.5 )
c
c        -kmax:  number of periods for which phase velocity
c                to be found
c        -t1: =0.0 the array t(i) i=1,kmax are read in
c             >0.0 initial period
c        -dt: period increment    t(i)=t1+(i-1)*dt will
c             be generated
c        -c1: initial phase velocity guess. better =0.8*b(1).
c
c        -t(i)  (only when t1=0.0)
c               kmax array of periods at which dispersion are
c               calculated.
c
c     7. if ipunch >=0
c        name of Love wave output file      (if idispl =1)
c        name of Rayleigh wave output file  (if idispr =1)
c
c     8. stop the program  (use -1,0)
c        or go back to 1 for other models.
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      common d(500),a(500),b(500),rho(500),mmax,ipunch
      common t(4096),c(500),mode,llw,twopi
   10 format(1h ,5x,4f10.4)
   20 format(1h ,15x,3f10.4)
   30 format('\f')
   40 format(/,1h ,19x,'crustal model  ',/1h )
```

```
   50 format(1h ,7x,'   thick     p-vel     s-vel     density'//)
      twopi=2.*3.141592653
  100 write(6,*) 'enter mmax(layers),mode(modes)'
      read(5,*)mmax,mode
c     mmax = number of layers to be read in, including
c            the halfspace.
c     mode = number of modes for which dispersion curves are
c            desired can be as large as 1000.
      if(mmax) 400,300,200
  200 continue
      l=mmax-1
c     d = thickness of layer in kilometers
c     a = compressional wave velocity in km/sec
c     b = transverse wave velocity in km/sec
c     rho = density in gm/cc
      write(6,*) 'enter d,a,b,rho'
      do 220 i=1,l
      read(5,*) d(i),a(i),b(i),rho(i)
  220 continue
      read(5,*) a(mmax),b(mmax),rho(mmax)
      write(6,30)
      write(6,40)
      write(6,50)
      llw=1
      if(b(1).le.0.0) llw=2
      do 240 i=1,l
  240 write(6,10) d(i),a(i),b(i),rho(i)
      write(6,20) a(mmax),b(mmax),rho(mmax)
  300 continue
      d(mmax)=0.0
      write(6,*) 'igphl,igphr,icut,idispl,idispr,ipunch'
c     igphl  gt 0 calls love wave dispersion plot
c     igphr  gt 0 calls rayleigh wave dispersion plot
c     icut   gt 0 calls search for love and rayleigh higher
c                 mode cutoff
c     idispl gt 0 love wave dispersion curve
c     idispr gt 0 rayl wave dispersion curve
c     ipunch eq 1  store and print the dispersion values
c            eq 0  store the dispersion values only
c            eq -1 print the dispersion values only
      read(5,*) igphl,igphr,icut,idispl,idispr,ipunch
      if(igphl.gt.0.or.igphr.gt.0) call gphdis(igphl,igphr)
      if(icut.gt.0) call cutoff
      if(idispl.gt.0.or.idispr.gt.0) call disper(idispl,idispr)
      go to 100
  400 continue
      stop
      end
c
c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine gphdis(igphl,igphr)
      doubleprecision wvno,omega
```

```
      common d(500),a(500),b(500),rho(500),mmax,ipunch
      common t(4096),c(500),mode,llw,twopi
      character*1 num(10),ks(60),kx(60)
      data num/'1','2','3','4','5','6','7','8','9',' '/
c     This subroutine graphically displays the sign of the
c     love or rayleigh wave period equation in the c-t plane.
c     The dispersion curve is the line of zeroes.
c     gphdis reads in up tp 59 different periods to form
c     abscissa of plot.
c     The ordinate varies from c1 to c2 in increments of dc
c     kk is the number of abscissa values
c     c1 is less than c2
c     dc is positive
   10 format('\f')
   20 format(5x,'period for abscissa of following graph'//)
   30 format(1x,12(i3,f7.2))
   40 format(19x,'plot of   love   function'//)
   50 format(19x,'plot of rayleigh function'//)
   60 format(2x,f8.4,59(a1,a1))
      write(6,*) 'gphdis-kk,cmin,cmax,dc'
      read(5,*)  kk,c1,c2,dc
      write(6,*) 'enter periods'
      read(5,*)  (t(i),i=1,kk)
      if(kk.gt.59) kk = 59
      if(dc.lt.0.0) dc = -dc
      if(c1.lt.c2) go to 100
      dum = c1
      c1 = c2
      c2 = dum
  100 continue
      write(6,10)
      write(6,20)
      write(6,30) (i,t(i),i=1,kk)
      do 700 ifunc = 1,2
      if(dc.eq.0.0) go to 700
      go to (200,300),ifunc
  200 if(igphl.le.0) go to 700
      write(6,10)
      write(6,40)
      go to 400
  300 if(igphr.le.0) go to 700
      write(6,10)
      write(6,50)
  400 cc = c2
  500 do 600 i=1,kk
      omega=dble(twopi)/dble(t(i))
      wvno=omega/dble(cc)
      del = dltar(wvno,omega,ifunc)
      l = mmax
      l10 = l/10
      l = l - l10*10
      kx(i)=num(l)
      ks(i)='+'
```

```
      if(del.lt.0.0) ks(i)='.'
      if(del.lt.0.0) kx(i)=' '
  600 continue
      write(6,60) cc,(ks(l),kx(l),l=1,kk)
      cc = cc - dc
      if(cc.ge.c1) go to 500
  700 continue
      return
      end
c
c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine cutoff
      double precision wvno,omega,cc
      common d(500),a(500),b(500),rho(500),mmax,ipunch
      common t(4096),c(500),mode,llw,twopi
c     kmax = number of cutoff periods to be found for phase
c            velocity c1.
c     t1 = initial period in search
c     dt = period increment  negative if  starting at high
c          period and going toward shorter period
c     c1 = phase velocity for which kmax cutoffs are being found
c     this routine finds both love and rayleigh cutoffs
   10 format('\f'//)
   20 format(///16x,'higher mode cutoff periods'//17x,
     1 '    love     ',23x,' rayleigh   '//2(11x,'period',10x,
     2 'ph vel')///)
   30 format(1h ,7x,f10.4,6x,f10.4,7x,f10.4,6x,f10.4)
      write(6,*) 'cutoff-enter kmax,t1,dt,c1'
      read(5,*) kmax,t1,dt,c1
      cc=dble(c1)
      write(6,10)
      write(6,20)
      tt=t1
      do 700 ifunc=1,2
      do 600 j=1,kmax
      tcut=0.0
      t1=tt
      omega=dble(twopi)/dble(t1)
      wvno=omega/cc
      del1 = dltar(wvno,omega,ifunc)
  100 continue
      t2 = t1 + dt
      if(t2.le.0.0) go to 700
      omega=dble(twopi)/dble(t2)
      wvno=omega/cc
      del2=dltar(wvno,omega,ifunc)
      if(sign(1.,del1).ne.sign(1.,del2)) go to 200
      t1 = t2
      del1 = del2
      go to 100
  200 if(abs (t1 - t2) - 0.0001) 500,500,300
  300 t3 = (t1 + t2) * 0.5
```

```
      omega=dble(twopi)/dble(t3)
      wvno=omega/cc
      del3 = dltar(wvno,omega,ifunc)
      if(sign(1.,del1).ne.sign(1.,del3)) go to 400
      t1 = t3
      del1 = del3
      go to 200
  400 t2 = t3
      del2 = del3
      go to 200
  500 continue
      tcut=0.5*(t1+t2)
      if(t2.le.0.0) go to 700
      kk=j
      if(ifunc.eq.1) t(j)=tcut
      if(ifunc.eq.2) t(j+500)=tcut
  600 continue
  700 continue
      write(6,30) (t(j),c1,t(j+500),c1,j=1,kk)
      return
      end
c
c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine disper(idispl,idispr)
      double precision omega0,omega1
      double precision eroot(500),cphase(500)
      common d(500),a(500),b(500),rho(500),mmax,ipunch
      common t(4096),c(500),mode,llw,twopi
      common/stor/ eroot(500),cphase(500),nlost,index,nroot1
      character*25 names
c     the root determination section is one of interval halving
c     once a zero crossing has been found.
c
c     the number of modes is allowed to be as large as 500,
c     and the number of periods as 4096.
c
c     t1 = initial starting period
c     kmax = number of period s for which phase velocity is to
c     be determined
c     if t1 = 0 program reads in array of t(i) periods instead
c     of computing them
c     dt = period increment. next period t2 = t1 + dt
c     c1 = initial phase velocity guess.  make sure it is
c     outside desired result
    5 format(a)
   10 format(1x,' improper initial value  no zero found ')
      write(6,*) 'enter kmax,t1,dt,c1 ; t1<=0 to read in array'
      read(5,*) kmax,t1,dt,cmin
      if(t1.gt.0.0) go to 100
      write(6,*) 'disper-enter kmax periods'
      read(5,*) (t(j),j=1,kmax)
      go to 200
```

```
  100 t(1) = t1
      do 150 i =2,kmax
  150 t(i) = t(i-1) + dt
  200 continue
      if(ipunch.lt.0) go to 400
      if(idispl.le.0) go to 300
      write(6,*) 'enter name of Love wave output file:'
      read(5,5) names
      open(1,file=names,status='new',form='unformatted')
      rewind 1
  300 continue
      if(idispr.le.0) go to 400
      write(6,*) 'enter name of Rayleigh wave output file:'
      read(5,5) names
      open(2,file=names,status='new',form='unformatted')
      rewind 2
  400 continue
      open(3,file='tmpsrf.d',status='scratch',form='unformatted')
      do 2000 ifunc=1,2
      ii = ifunc
      nlost = 0
      index = 0
      do 500 i=1,500
      c(i)=0.0
      eroot(i)=0.0
      cphase(i)=0.0
  500 continue
      rewind 3
      if(ifunc.eq.1.and.idispl.le.0) go to 2000
      if(ifunc.eq.2.and.idispr.le.0) go to 2000
      nroot1=1000
      do 1800 k = 1,kmax
      t1 = t(k)
      omega1=dble(twopi)/dble(t1)
      if(k.gt.1) omega0=dble(twopi)/dble(t(k-1))
      index=index+1
cxxxxxxx
      kmode=0
      call poles(ii,omega0,omega1,cmin)
      kmode=nroot1
cxxxxxxxx
      if(k.eq.1.and.kmode.eq.0) write(6,10)
      if(k.eq.1.and.kmode.eq.0) go to 2000
      if(k.eq.1) lmode=kmode
      write(3) ifunc,kmode,t1
      do 1750 i=1,kmode
      cc=omega1/eroot(i)
      write(3) cc
 1750 continue
      i=-1
      tmp=0.0
      if(k.eq.kmax) write(3) i,i,tmp
 1800 continue
```

```
      call output(ifunc,lmode)
      if(ipunch.lt.0) go to 2000
      write(ifunc) mmax
      do 1810 i=1,mmax
      write(ifunc) d(i),a(i),b(i),rho(i)
 1810 continue
      write(ifunc) kmax
      rewind 3
 1850 continue
      read(3) ifun,kmode,t1
      write(ifunc) ifun,kmode,t1
      if(ifun.le.0) go to 1900
      do 1860 i=1,kmode
      read(3) c0
      write(ifunc) c0
 1860 continue
      go to 1850
 1900 continue
      close(ifunc)
 2000 continue
      close(3,status='delete')
      return
      end
c
c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine output(ifunc,lmode)
      common d(500),a(500),b(500),rho(500),mmax,ipunch
      common t(4096),c(500),mode,llw,twopi
   10 format(///15x,'Love wave      mode #',i3//21x,
     1 '    period          phase vel')
   20 format(1h ,12x,i3,1x,f15.10,2x,f15.10)
   30 format(///14x,'Rayleigh wave      mode #',i3//21x,
     1 '    period            phase vel')
   50 format('\f')
      write(6,50)
      write(6,*) '        number of modes at the lowest period = ',
     *                lmode
      nm=0
      c1=b(mmax)
  100 continue
      nm=nm+1
      if(nm.gt.lmode) go to 600
      if(ipunch.ne.0.and.ifunc.eq.1) write(6,10) nm
      if(ipunch.ne.0.and.ifunc.eq.2) write(6,30) nm
      rewind 3
      nt=0
      do 200 i=1,500
      c(i)=0.0
  200 continue
      kk=0
  250 continue
      read(3) ifun,kmode,t0
```

```
      if(ifun) 450,300,300
  300 continue
      nt=nt+1
      do 400 i=1,kmode
      read(3) c0
      if(i.ne.nm) go to 400
      c(nt)=c0
      kk=nt
  400 continue
      go to 250
  450 continue
      if(ipunch.eq.0) go to 100
      write(6,20) (i,t(i),c(i),i=1,kk)
      go to 100
  600 continue
      return
      end
c
c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine poles(ifunc,omega0,omega,cmin)
      double precision eroot(500),cphase(500)
      double precision omega0,omega,dk,domega,dk0,tadd
      double precision wvmn,wvmx,c1,c2,c3,c4,s,t,vphase,dc
      double precision wvm(500),c10
      common d(500),a(500),b(500),rho(500),mmax,ipunch
      common perd(4096),c(500),mode,llw,twopi
      common/stor/ eroot(500),cphase(500),nlost,index,nroot1
c     this routine finds the roots of period equation using
c     regular halving method to initialize the first two
c     frequencies, then followed by jumping method.
c
      epi = 1.e-8
      freq = omega/twopi
      wvmx = omega/dble(cmin)
      wvmn = omega/dble(b(mmax))
      nmx=200 + (freq*200)
      dk = (wvmx-wvmn)/nmx
      if(nlost.eq.1001) go to 2000
      if(index.gt.2) go to 3000
 2000 continue
c     find the poles using the regular halving method
c     nmx is chosen for a 40 km crustal model. for shallower
c     thickness a proportionately smaller nmx can be used
c     search for roots of period equation
      if(index.gt.2) write(6,*) '-at period=',twopi/omega,
     *                 ' return halving method.'
      do 80 i=llw,mmax
      wvm(i)=omega/dble(b(i))
   80 continue
      nroot = 0
      c2 = wvmx
      del2 = dltar(c2,omega,ifunc)
```

```
      lyr=llw
      jj=1
      do 500 i=1,nmx
      jj=jj-1
      if(jj.ne.0) go to 500
      c10 = wvmx-i*dk
      if(i.eq.nmx) c10=wvmn+0.01*dk
      jj = 1
      kk = 1
      if(c10.gt.wvm(lyr)) go to 90
c     kk and jj represent a denser seraching when phase velocity
c     = S wave velocity. Their values can be changed as kk=3*lyr
c     jj=8.
      kk = 10.0*(a(lyr+1)/a(lyr))+0.5*lyr
      lyr = lyr+1
      jj = 5
   90 continue
      dk0 = dk/float(kk)
      do 400 j=1,jj
      do 400 k=1,kk
      if(nroot.eq.mode) go to 510
      jk = kk*(j-1)+k
      c1 = c10+dk-dk0*float(jk)
      if(c1.lt.wvmn) go to 510
      del1 = dltar(c1,omega,ifunc)
      if(sign(1.0,del1)*sign(1.0,del2).ge.0.0) go to 350
      nroot = nroot + 1
      c4 = c2
      del4 = del2
      do 200 ii=1,15
      c3 = 0.5*(c1+c4)
      del3 = dltar(c3,omega,ifunc)
      if(sign(1.0,del1)*sign(1.0,del3).ge.0.0) go to 100
      del4 = del3
      c4 = c3
      go to 150
  100 del1 = del3
      c1 = c3
  150 continue
      if(abs(c4-c1).lt.epi) go to 250
  200 continue
  250 continue
      c3 = 0.5*(c1+c4)
      if(index.eq.1) cphase(nroot)=omega/c3
      if(nlost.eq.1) cphase(nroot)=omega/c3
      eroot(nroot) = c3
  350 c2 = c1
      del2 = del1
  400 continue
  500 continue
  510 continue
      nlost=nlost+1000
      go to 1250
```

```
 3000 continue
c      the jumping method.
c      if this method fails, the control will return
c      to the regular halving method.
c      xxxxxx
       nroot = 0
       if(nroot1.eq.0) go to 1250
       dk0=0.25*dk
       domega = omega0-omega
       eroot(nroot1+1)=wvmn
       if(nroot1.eq.mode) eroot(nroot1+1)=eroot(nroot1)-5.*dk
       nlost  = 0
       do 1200 i=1,nroot1
       s      = eroot(i)
       vphase = omega0/s
       t      = -domega/vphase
       dc     = vphase-cphase(i)
       tadd   = -dc*s/vphase
       dc     = s-eroot(i+1)
       c1 = dabs(tadd)/tadd
       if(i.eq.1) go to 520
       if(dabs(tadd).lt.dk0) tadd=c1*dk0
       if(dabs(tadd).gt.dc) tadd=c1*0.5*dc
  520  continue
       c2     = s+t
       nctrl  = 5
       itrig  = 0
       if(i.gt.1.and.c2.ge.eroot(i-1)) itrig=1
       if(itrig.eq.0) go to 530
       if(dabs(tadd).lt.dk0) tadd=c1*dk0
c      If roots are duplicated, a correction might be done here.
       c2 = eroot(i-1)-0.1*abs(dc)
       nctrl = 10
  530  continue
       c1     = c2+tadd
       ihalf  = 20
       ncont  = 0
  550  if(c2.le.wvmn) go to 1250
       del2 = dltar(c2,omega,ifunc)
       ntest  = 0
  600  if(c1.le.wvmn) go to 800
       del1 = dltar(c1,omega,ifunc)
       if(sign(1.0,del1)*sign(1.0,del2).le.0.0) go to 850
       ntest  = ntest+1
       if(ntest.ge. nctrl) go to 650
       del2   = del1
       c2     = c1
       c1     = c1+tadd
       go to 600
  650  ncont  = ncont+1
       go to (700,720,750),ncont
  700  continue
c      This is another kind of jumping procedure, which is
```

```
c     a remedy when the first jumping method fails.
      ihalf  = 20
      tadd   = -dc/20.d+00
      c2     = s+t
      nctrl  = 60
      if(i.eq.1) nctrl = 15
      if(itrig.eq.1) c2=eroot(i-1)-0.7*dk
      go to 740
  720 continue
      if(itrig.eq.1) go to 750
      if(i.eq.mode) go to 1200
c     This is the third kind of jumping procedure.
      tadd   = 0.25*t
      c2     = s+tadd
      if(i.eq.1) go to 730
      if(c2.lt.eroot(i-1)) go to 730
      tadd = -abs(s+t-eroot(i-1))/4.
      c2 = eroot(i-1)+tadd
  730 continue
      nctrl  = 3
      ihalf  = 100
  740 c1     = c2+tadd
      go to 550
  750 nlost  = nlost+1
c     If all jumping methods fail, it goes back to the regular
c     halving method.
      go to 2000
  800 c1     = wvmn+0.01*dk
      del1 = dltar(c1,omega,ifunc)
      if(sign(1.0,del1)*sign(1.0,del2).le.0.0) go to 850
      go to 1250
  850 c4     = c2
      del4   = del2
      do 1000 ii=1,ihalf
      c3     = 0.5*(c1+c4)
      del3 = dltar(c3,omega,ifunc)
      if(sign(1.0,del1)*sign(1.0,del3) .ge. 0.0) go to 900
      del4   = del3
      c4     = c3
      go to 950
  900 del1   = del3
      c1     = c3
  950 continue
      if(abs(c4-c1).lt.epi) go to 1050
 1000 continue
 1050 continue
      c3 = 0.5*(c1+c4)
      nroot = nroot+1
      eroot(nroot)  = c3
      cphase(nroot) = vphase
 1200 continue
 1250 continue
      if(nroot.gt.nroot1) nroot=nroot1
```

```
      nroot1=nroot
      return
      end
c
c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      function dltar(wvno,omega,kk)
      double precision wvno,omega
      common d(500),a(500),b(500),rho(500),mmax,ipunch
      common t(4096),c(500),mode,llw,twopi
c
      go to (10,20),kk
c     love wave period equation
   10 dltar = dltar1(wvno,omega)
      return
c     rayleigh wave period equation
   20 dltar = dltar4(wvno,omega)
      return
      end
c
c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      function dltar1(wvno,omega)
      double precision xnor,ynor,wvno,omega
      double precision wvno2,xkb,rb,e1,e2,xmu,q,rho1,beta1
      double precision sinq,cosq,y,z,exqp,exqm,e10,e20
c     haskell-thompson love wave formulation from halfspace
c     to surface.
      common d(500),a(500),b(500),rho(500),mmax,ipunch
      common d1(4096),d2(500),mode,llw,twopi
      wvno2=wvno*wvno
      beta1=dble(b(mmax))
      rho1=dble(rho(mmax))
      xkb=omega/beta1
      rb = dsqrt(dabs(wvno2-xkb*xkb))
      e1=rho1*rb
      e2=1.d+00/(beta1*beta1)
      mmm1 = mmax - 1
      do 600 m=mmm1,llw,-1
      beta1=dble(b(m))
      rho1=dble(rho(m))
      xmu=rho1*beta1*beta1
      xkb=omega/beta1
      rb = dsqrt(dabs(wvno2-xkb*xkb))
      q = dble(d(m))*rb
      if(wvno-xkb) 100,200,300
  100 sinq = dsin(q)
      y = sinq/rb
      z = -rb*sinq
      cosq = dcos(q)
      go to 500
  200 cosq=1.0d+00
      y=dble(d(m))
```

```
      z=0.0d+00
      go to 500
  300 continue
      if(q.gt.40.) go to 400
      exqp=dexp(q)
      exqm=1./exqp
      sinq = (exqp-exqm)/2.
      y = sinq/rb
      z = sinq*rb
      cosq = (exqp + exqm)/2.
      go to 500
  400 continue
      y = 0.5d+00/rb
      z = 0.5d+00*rb
      cosq = 0.5d+00
  500 continue
      e10=e1*cosq+e2*xmu*z
      e20=e1*y/xmu+e2*cosq
      xnor=dabs(e10)
      ynor=dabs(e20)
      if(ynor.gt.xnor) xnor=ynor
      if(xnor.lt.1.d-30) xnor=1.0d+00
      e1=e10/xnor
      e2=e20/xnor
  600 continue
      dltar1=e1
      return
      end
c
c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      function dltar4(wvno,omga)
      double precision e(5),ee(5),ca(5,5)
      double precision wvno,omga,omega,wvno2,rho1
      double precision xka,xkb,ra,rb,t,gam,gammk,gamm1
      double precision exa,p,q,dpth,w,w0,cosp,cr
      double precision a0,cpcq,cpy,cpz,cqw,cqx,xy,xz,wy,wz
      common d(500),a(500),b(500),rho(500),mmax,ipunch
      common per(4096),c(500),mode,llw,twopi
      common/ovrflw/ a0,cpcq,cpy,cpz,cqw,cqx,xy,xz,wy,wz
      omega=omga
      if(omega.lt.1.0d-5) omega=1.0d-5
      wvno2=wvno*wvno
      xka=omega/dble(a(mmax))
      xkb=omega/dble(b(mmax))
      ra=dsqrt(dabs(wvno2-xka*xka))
      rb=dsqrt(dabs(wvno2-xkb*xkb))
      t = dble(b(mmax))/omega
      gammk = 2.*t*t
      gam = gammk*wvno2
      gamm1 = gam - 1.
      rho1=dble(rho(mmax))
      e(1)=rho1*rho1*(gamm1*gamm1-gam*gammk*ra*rb)
```

```
      e(2)=-rho1*ra
      e(3)=rho1*(gamm1-gammk*ra*rb)
      e(4)=rho1*rb
      e(5)=wvno2-ra*rb
c     matrix multiplication from bottom layer upward
      mmm1 = mmax-1
      do 500 m = mmm1,llw,-1
      xka = omega/dble(a(m))
      xkb = omega/dble(b(m))
      t = dble(b(m))/omega
      gammk = 2.*t*t
      gam = gammk*wvno2
      ra = dsqrt(dabs(wvno2-xka*xka))
      rb = dsqrt(dabs(wvno2-xkb*xkb))
      dpth=dble(d(m))
      rho1=dble(rho(m))
      p=ra*dpth
      q=rb*dpth
      beta=b(m)
      call var(p,q,ra,rb,wvno,xka,xkb,dpth,w,cosp,exa,beta)
      call dnka(ca,wvno2,gam,gammk,rho1)
      do 200 i=1,5
      cr=0.0d+00
      do 100 j=1,5
      cr=cr+e(j)*ca(j,i)
  100 continue
      ee(i)=cr
  200 continue
      call normc(ee,exa)
      do 300 i = 1,5
      e(i)=ee(i)
  300 continue
  500 continue
      w0 = 0.0
      if(llw.eq.1) go to 600
      xka = omega/dble(a(1))
      ra = dsqrt(dabs(wvno2-xka*xka))
      dpth=dble(d(1))
      rho1=dble(rho(1))
      p = ra*dpth
      beta = b(1)
      call var(p,q,ra,rb,wvno,xka,xkb,dpth,w,cosp,exa,beta)
      w0=-rho1*w/cosp
  600 continue
      dltar4=e(1)+w0*e(2)
      return
      end
c
c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine var(p,q,ra,rb,wvno,xka,xkb,dpth,w,cosp,exa,beta)
      double precision p,q,ra,rb,wvno,xka,xkb,dpth
      double precision w,x,y,z,cosp,cosq,sinp,sinq
```

```
      double precision exa,expp,expm,exqp,exqm
      double precision a0,cpcq,cpy,cpz,cqw,cqx,xy,xz,wy,wz
      common/ovrflw/   a0,cpcq,cpy,cpz,cqw,cqx,xy,xz,wy,wz
      exa=0.0d+00
      a0=1.0d+00
      if(wvno-xka) 100,200,300
  100 sinp=dsin(p)
      w=sinp/ra
      x=-ra*sinp
      cosp=dcos(p)
      go to 500
  200 cosp=1.0d+00
      w=dpth
      x=0.0d+00
      go to 500
  300 if(p.gt.40.0) go to 400
      expp=dexp(p)
      expm=1./expp
      sinp=(expp-expm)*0.5d+00
      cosp=(expp+expm)*0.5d+00
      w=sinp/ra
      x=ra*sinp
      go to 500
  400 exa=p
      x=ra*0.5d+00
      w=0.5d+00/ra
      cosp=0.5d+00
      a0=0.0d+00
      if(exa.lt.70.0) a0=1./dexp(exa)
  500 continue
      if(beta.lt.1.e-5) return
      if(wvno-xkb) 600,700,800
  600 sinq=dsin(q)
      z=-rb*sinq
      y=sinq/rb
      cosq=dcos(q)
      go to 1000
  700 cosq=1.0d+00
      y=dpth
      z=0.0d+00
      go to 1000
  800 if(q.gt.50.0) go to 900
      exqp=dexp(q)
      exqm=1./exqp
      sinq=(exqp-exqm)*0.5d+00
      cosq=(exqp+exqm)*0.5d+00
      y=sinq/rb
      z=rb*sinq
      go to 1000
  900 y=0.5d+00/rb
      z=0.5d+00*rb
      cosq=0.5d+00
      exa=exa+q
```

```
      a0=0.0d+00
      if(exa.lt.70.0) a0=1./dexp(exa)
 1000 continue
      cpcq=cosp*cosq
      cpy=cosp*y
      cpz=cosp*z
      xy=x*y
      xz=x*z
      wy=w*y
      wz=w*z
      cqw=cosq*w
      cqx=cosq*x
      return
      end
c
c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine normc(ee,ex)
c     This routine is an important step to control over- or
c     underflow.
c     The Haskell or Dunkin vectors are normalized before
c     the layer matrix stacking.
c     Note that some precision will be lost during normalization.
c
      double precision ee(5),ex,t1,t2
      ex = 0.0d+00
      t1 = 0.0d+00
      do 10 i = 1,5
      if(dabs(ee(i)).gt.t1) t1 = dabs(ee(i))
   10 continue
      if(t1.lt.1.d-30) t1=1.d+00
      do 20 i =1,5
      t2=ee(i)
      t2=t2/t1
      ee(i)=t2
   20 continue
      ex=dlog(t1)
      return
      end
c
c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine dnka(ca,wvno2,gam,gammk,rho)
      double precision ca(5,5),wvno2,gam,gammk,rho
      double precision gamm1,twgm1,gmgmk,gmgm1,gm1sq,rho2,t
      double precision a0,cpcq,cpy,cpz,cqw,cqx,xy,xz,wy,wz,a0pq
      common/ ovrflw / a0,cpcq,cpy,cpz,cqw,cqx,xy,xz,wy,wz
      gamm1 = gam-1.
      twgm1=gam+gamm1
      gmgmk=gam*gammk
      gmgm1=gam*gamm1
      gm1sq=gamm1*gamm1
      rho2=rho*rho
```

```
      aOpq=aO-cpcq
      ca(1,1)=cpcq-2.*gmgm1*aOpq-gmgmk*xz-wvno2*gm1sq*wy
      ca(1,2)=(wvno2*cpy-cqx)/rho
      ca(1,3)=-(twgm1*aOpq+gammk*xz+wvno2*gamm1*wy)/rho
      ca(1,4)=(cpz-wvno2*cqw)/rho
      ca(1,5)=-(2.*wvno2*aOpq+xz+wvno2*wvno2*wy)/rho2
      ca(2,1)=(gmgmk*cpz-gm1sq*cqw)*rho
      ca(2,2)=cpcq
      ca(2,3)=gammk*cpz-gamm1*cqw
      ca(2,4)=-wz
      ca(2,5)=ca(1,4)
      ca(4,1)=(gm1sq*cpy-gmgmk*cqx)*rho
      ca(4,2)=-xy
      ca(4,3)=gamm1*cpy-gammk*cqx
      ca(4,4)=ca(2,2)
      ca(4,5)=ca(1,2)
      ca(5,1)=-(2.*gmgmk*gm1sq*aOpq+gmgmk*gmgmk*xz+
     *             gm1sq*gm1sq*wy)*rho2
      ca(5,2)=ca(4,1)
      ca(5,3)=-(gammk*gamm1*twgm1*aOpq+gam*gammk*gammk*xz+
     *             gamm1*gm1sq*wy)*rho
      ca(5,4)=ca(2,1)
      ca(5,5)=ca(1,1)
      t=-2.*wvno2
      ca(3,1)=t*ca(5,3)
      ca(3,2)=t*ca(4,3)
      ca(3,3)=aO+2.*(cpcq-ca(1,1))
      ca(3,4)=t*ca(2,3)
      ca(3,5)=t*ca(1,3)
      return
      end
c********
```

```
c     dpsrf81.f
c
c     f77 -i -I2 dpsrf.f -o dpsrf -lcalcompI2
c
c     plot dispersion curve
c     get data from surface81
c
      dimension t(2048),v(2048),mode(500)
      dimension d(500),a(500),b(500),rho(500)
      character*20 names
      character*4 yorn
 5    format(a)
 15   format(4f12.5)
      pi=2.*3.141592653
      write(6,*) ' '
      write(6,*) 'Rayleigh(1) or Love(2):'
      read(5,*) llrr
      write(6,*) ' '
      write(6,*) 'enter the input file name (from surface81):'
      read(5,5) names
      open(1,file=names,status='old',form='unformatted')
      write(6,*)
     *   'how many modes and which modes wanted? (3,1,3,5)'
      write(6,*) '(if all modes wanted, answer 1,0 )'
      read(5,*) nmode,(mode(i),i=1,nmode)
      if(mode(1).ne.0) go to 40
      nmode=500
      do 30 i=1,500
      mode(i)=i
 30   continue
 40   continue
      write(6,*) 'plot  C-perd(1)  C-freq(2)  K-C(3)    C-K(4)'
      write(6,*) '      K-freq(5)  freq-K(6):'
      read(5,*) ipg
      call plots(0,0,7)
      write(6,*) 'enter ipen:'
      read(5,*) ipen
      call newpen(ipen)
      yorn='y'
      call plot(2.5,2.1,-3)
      write(6,*) ' '
      write(6,*) 'model:'
      do 600 kk=1,nmode
      rewind 1
      read(1) nmax
      do 50 ii=1,nmax
      read(1) d(ii),a(ii),b(ii),rho(ii)
 50   continue
      if(kk.eq.1) write(6,15) (d(i),a(i),b(i),rho(i),i=1,nmax)
      read(1) nper
 100  continue
      ident=mode(kk)
      nt=0
```

```
      itrig=0
  200 continue
      read(1) ifunc,kmode,t1
      if(ifunc.lt.0) go to 400
      if(kmode.le.0) go to 200
      nt=nt+1
      do 300 i=1,kmode
      read(1) c0
      wvno0=2.*3.141592653/(c0*t1)
      if(i.ne.ident) go to 300
      itrig=1
      nn=nt
      t(nn)=t1
      v(nn)=c0
  300 continue
      go to 200
  400 continue
      if(itrig.eq.0.and.nmode.eq.500) go to 600
      do 500 i=1,nn
      go to (500,482,487,488,485,486),ipg
  482 t(i)=1./t(i)
      go to 500
  485 t(i)=1./t(i)
      v(i)=pi*t(i)/v(i)
      go to 500
  486 tmp=1./t(i)
      t(i)=pi*tmp/v(i)
      v(i)=tmp
      go to 500
  487 tmp=v(i)
      v(i)=pi/(t(i)*v(i))
      t(i)=tmp
      go to 500
  488 t(i)=pi/(t(i)*v(i))
  500 continue
      call disp(yorn,itrig,kk,ipg,ident,nn,t,v)
  600 continue
      call plot(0.0,8.0,999)
      close(1)
      close(2)
      write(6,*) 'job finished'
      stop
      end
c
      subroutine disp(yorn,itrig,kk,ipg,ident,nn,x,y)
      dimension x(1),y(1)
      character*4 yorn,nory
      character*11 alpha(2,6),alp,bet
      data alpha/
     *  'C  (km/sec)','T  (sec)   ','C  (km/sec)','f  (Hz)    ',
     *  'K  (rad/km)','C  (km/sec)','C  (km/sec)','K  (rad/km)',
     *  'K  (rad/km)','f  (Hz)    ','f  (Hz)    ','K  (rad/km)' /
      if(itrig.eq.0)
```

```
*    write(6,*) 'mode ',ident,' is not generated.'
     if(itrig.eq.0) return
     alp=alpha(1,ipg)
     bet=alpha(2,ipg)
     if(yorn.eq.'n') go to 100
     yorn='n'
     xlen=6.0
     ylen=4.5
     xmin=1.e+37
     xmax=-1.e+37
     ymin=1.e+37
     ymax=-1.e+37
     do 80 i=1,nn
     if(x(i).ge.xmax) xmax=x(i)
     if(x(i).le.xmin) xmin=x(i)
     if(y(i).ge.ymax) ymax=y(i)
     if(y(i).le.ymin) ymin=y(i)
80   continue
     write(6,*) 'ymin=',ymin,' ymax=',ymax
     if(ipg.ge.9) write(6,*) '  -y value in log sacle'
     write(6,*) 'xmin=',xmin,' xmax=',xmax
     write(6,*) ' '
     write(6,*) 'enter ymin,ymax,yinc,xmin,xmax,xinc'
     if(ipg.ge.9) write(6,*) '       -y should be integer.'
     read(5,*) y0,y1,yinc,x0,x1,xinc
     xinut=(x1-x0)/xlen
     yinut=(y1-y0)/ylen
100  continue
     if(kk.ne.1) go to 500
     write(6,*) 'plot axis? (y/n)'
     read(5,5) nory
5    format(a)
     write(6,*) ' '
     write(6,*) 'wait.  It is processing.'
     if(nory.ne.'y') go to 500
     call plot(xlen,0.0,2)
     call plot(xlen,ylen,2)
     call plot(0.0,ylen,2)
     call plot(0.0,0.0,2)
     call plot(0.0,-0.05,2)
     if(x0.lt.0.0) xshif=-0.25
     if(x0.ge.0.0) xshif=-0.13
     if(x0.ge.10.0) xshif=-0.23
     if(xinc.ge.0.1) call number(xshif,-0.17,0.1,x0,0.0,1)
     if(xinc.lt.0.1)
    *call number(xshif-0.04,-0.16,0.09,x0,0.0,2)
     grid=xinc/xinut
     xi=1.
200  xx=xi*grid
     if(abs(xx).gt.xlen+0.02) go to 250
     call plot(xx,0.0,3)
     call plot(xx,-0.05,2)
     xsym=x0+xinc*xi
```

```
      if(xsym.lt.0.0) xshif=-0.25
      if(xsym.ge.0.0) xshif=-0.13
      if(xsym.ge.10.0) xshif=-0.23
      if(xsym.ge.100.0) xshif=-0.33
      if(xinc.ge.0.1)
     *call number(xx+xshif,-0.17,0.1,xsym,0.0,1)
      if(xinc.lt.0.1)
     *call number(xx+xshif-0.04,-0.16,0.09,xsym,0.0,2)
      xi = xi+1.
      go to 200
  250 continue
      call symbol(2.5,-0.4,0.16,bet,0.0,11)
      call plot(0.0,0.0,3)
      call plot(-0.05,0.0,2)
      if(ipg.ge.9) go to 260
      if(y0.lt.0.0) xshif=-0.45
      if(y0.ge.0.0) xshif=-0.35
      if(y0.ge.10.0) xshif=-0.45
      if(yinc.ge.0.1)
     *call number(xshif,-0.05,0.1,y0,0.0,1)
      if(yinc.lt.0.1)
     *call number(xshif-0.03,-0.05,0.09,y0,0.0,2)
      go to 270
  260 continue
      call number(-0.34,-0.06,0.1,10.0,0.0,-1)
      call number(-0.16,0.02,0.06,y0,0.0,-1)
  270 continue
      grid=yinc/yinut
      xi=1.
  300 xx=xi*grid
      if(abs(xx).gt.ylen+0.02) go to 400
      call plot(0.0,xx,3)
      call plot(-0.05,xx,2)
      call plot(0.0,xx,3)
      xsym=y0+yinc*xi
      if(ipg.ge.9) go to 310
      if(xsym.lt.0.0) xshif=-0.45
      if(xsym.ge.0.0) xshif=-0.35
      if(xsym.ge.10.0) xshif=-0.45
      if(xsym.ge.100.0) xshif=-0.55
      if(yinc.ge.0.1)
     *call number(xshif,xx-0.05,0.1,xsym,0.0,1)
      if(yinc.lt.0.1)
     *call number(xshif-0.04,xx-0.05,0.09,xsym,0.0,2)
      go to 320
  310 continue
      call number(-0.34,xx-0.06,0.1,10.0,0.0,-1)
      call number(-0.16,xx+0.02,0.06,xsym,0.0,-1)
  320 continue
      xi = xi+1.
      go to 300
  400 continue
      call symbol(-0.55,1.5,0.16,alp,90.0,11)
```

```
 500 continue
     do 550 i=1,nn
     if(y(i).gt.y1) y(i)=y1
     if(x(i).gt.x1) x(i)=x1
     if(y(i).lt.y0) y(i)=y0
     if(x(i).lt.x0) x(i)=x0
 550 continue
     x(nn+1)=x0
     x(nn+2)=xinut
     y(nn+1)=y0
     y(nn+2)=yinut
     call line(x,y,nn,1,0,0)
     return
     end
***********
```

```
c     reigen81.f
c
c     f77 -i -I2 reigen81.f -o reigen81
c
c     This program calculates the eigenfunctions of
c     Rayleigh wave for any plane multi-layered model.
c
c     Body wave Q model can be included.
c
c     The propagator-matrix, instead of numerical-integration
c     method is used, in which the Haskell rather than
c     Harkrider formalisms are concered.
c
c     Such a revision was developed to cover a large range of
c     frequencies, say 200 Hz, and to improve the calculation
c     efficiency. The layer thickness is not limitted.
c
c     For the sake of space saving, only the values of
c     eigenfunctions at the source depths are stored.
c     However, several files with different source depths
c     can be set up.
c
c     -Oct 10, 1981
c
      double precision sumi0,sumi1,sumi2,sumi3
      dimension nos(100),dphs(100),dphq(100),qa(100),qb(100)
      dimension depth(100)
      common/model/  d(100),a(100),b(100),rho(100),qa1(100),
     *               qb1(100),xmu(100),xlam(100),mmax,ll
      common/eigfun/ ur(100),uz(100),tz(100),tr(100),uu0(4),
     *               dcda(100),dcdb(100),dcdr(100)
      common/sumi/   sumi0,sumi1,sumi2,sumi3,flagr,are,ugr
      character*1 dd
      character*50 names
5     format(a)
10    format(/2x,'M',4x,'DPTH',2x,'   D  ',3x,'   A  ',3x,'   B  ',
     *       3x,' RHO',3x,' QA  ',4x,' QB  ',4x,' MU  ',3x,'LMDA')
20    format(i3,1x,5f7.2,2f8.2,2f7.2)
30    format(3x,'-source is on the top of this layer.  ',
     *       '  source depth=',6.2)
40    format(i3,1x,f7.2,7x,3f7.2,2f8.2,2f7.2)
50    format(3x,'-source is inside the half space.',
     *       '  source depth=',f6.2)
60    form at('at the source depth = ',f6.2,'  km')
      write(6,*)
     *  'enter the input file name: (from surface81)'
      read(5,5) names
      open(1,file=names,status='old',form='unformatted')
      rewind 1
      open(3,file='tmp.1',status='scratch',form='unformatted')
      rewind 3
c
c     enter source depths and Q-model:
```

```
c
      write(6,*) 'enter the source depths:'
      write(6,*)
     *   '(no. of source depths, srcdph(1), srcdph(2), ...)'
      write(6,*)
     *   '*** if no  of source depths is negative, no output'
      write(6,*) '    files will be generated. ***'
c     This can be used as dividing layers.
      read(5,*) kks,(dphs(i),i=1,iabs(kks))
      ks=abs(kks)
      write(6,*) 'enter Q-model here(1), from a file(2),'
      write(6,*) 'or NOT considered(3):'
      read(5,*) kk
      kq=1
      go to (100,110,170),kk
  100 continue
      write(6,*)
     *   'enter d(i),Qa(i),Qb(i)  use d(i)=0.0 for halfspace'
      icode=5
      go to 120
  110 continue
      write(6,*) 'enter the name of the file storing Q-model:'
      read(5,5) names
      open(2,file=names,status='old',form='formatted')
      rewind 2
      icode=2
  120 continue
      i=1
      base=0.0
  140 continue
      read(icode,*) dq0,qa(i),qb(i)
      if(dq0.le.0.0) go to 150
      base=base+dq0
      dphq(i)=base
      i=i+1
      go to 140
  150 kq=i
      do 160 i=1,100
      qa1(i)=qa(kq)
      qb1(i)=qb(kq)
  160 continue
  170 continue
      write(6,*) 'Store the derivatives? (y/n):'
      read(5,5) dd
      if(dd.eq.'n') go to 175
      write(6,*) 'enter the output file name for derivatives:'
      read(5,5) names
      open(9,file=names,status='new',form='unformatted')
      rewind 9
  175 continue
      do 180 i=1,100
      depth(i)=10000000.0
  180 continue
```

```
c
c     obtain the earth model:
c
      read(1) mmax
      write(6,10)
      base=0.0
      depth(1)=0.0
      do 190 i=1,mmax
      read(1) d(i),a(i),b(i),rho(i)
      base=base+d(i)
      depth(i+1)=base
      xmu(i)=rho(i)*b(i)*b(i)
      xlam(i)=rho(i)*(a(i)*a(i)-2.*b(i)*b(i))
 190  continue
c
c     insert the Q-model into the velocity model.
c     insert the source depth as an interface of layers.
c
      kqs=kq+ks-1
      do 300 k0=1,kqs
      dph0=dphq(k0)
      is=k0-kq+1
      if(k0.ge.kq) dph0=dphs(is)
      do 200 i=1,mmax
      k=i
      if(dph0.eq.depth(i)) go to 250
      if(dph0.gt.depth(i).and.dph0.lt.depth(i+1)) go to 210
 200  continue
 210  k1=k+1
      dphk1=depth(k1)
      do 220 i=mmax,k,-1
      i1=i+1
      d(i1)=d(i)
      a(i1)=a(i)
      b(i1)=b(i)
      rho(i1)=rho(i)
      xmu(i1)=xmu(i)
      xlam(i1)=xlam(i)
      depth(i1)=depth(i)
      if(k0.ge.kq) qa1(i1)=qa1(i)
      if(k0.ge.kq) qb1(i1)=qb1(i)
 220  continue
      d(k)=dph0-depth(k)
      d(k1)=dphk1-dph0
      depth(k1)=depth(k)+d(k)
      mmax=mmax+1
      if(k0.ge.kq) go to 240
      if(k0.eq.1) ns=1
      do 230 j=ns,k
      qa1(j)=qa(k0)
      qb1(j)=qb(k0)
 230  continue
 240  ns=k1
```

```
      go to 280
250   continue
      if(k0.ge.kq) go to 270
      if(k0.eq.1) ns=1
      do 260 i=ns,k-1
      qa1(i)=qa(k0)
      qb1(i)=qb(k0)
260   continue
270   ns=k
280   nos(is)=ns
300   continue
      j=1
      do 310 i=1,mmax-1
      write(6,20) i,depth(i),d(i),a(i),b(i),rho(i),qa1(i),
     *               qb1(i),xmu(i),xlam(i)
      if(i.ne.nos(j)) go to 310
      write(6,30) dphs(j)
      j=j+1
310   continue
      i=mmax
      write(6,40)
     *  i,depth(i),a(i),b(i),rho(i),qa1(i),qb1(i),xmu(i),xlam(i)
      if(nos(j).eq.mmax) write(6,50) dphs(j)
      if(dd.eq.'y')
     *   write(9) mmax,(d(i),a(i),b(i),rho(i),i=1,mmax)
      write(6,*) ' '
      write(6,*) 'wait.'
      l1=1
      if(b(1).le.0.0) l1=2
      read(1) nper
 400  continue
c
c     read in the dispersion values.
c
      read(1) ifunc,mode,t
      write(3) ifunc,mode,t
      if(dd.eq.'y') write(9) ifunc,mode,t
      if(ifunc.lt.0) go to 700
      if(mode.le.0) go to 400
      do 600 k=1,mode
      read(1) c
c
c     main part.
c
      omega=6.2831853/t
      wvno=omega/c
      call svfunc(omega,wvno)
      call energy(omega,wvno)
      omega2=omega*omega
      wvomg2=wvno*omega2
      do 450 i=l1,mmax
      ur(i)=ur(i)*wvno
      tz(i)=tz(i)*omega2
```

```
      tr(i)=tr(i)*wvomg2
  450 continue
      if(kk.ne.3) call gammap(omega,wvno,gamma)
      if(dd.eq.'n') go to 510
c     output the derivatives.
      xi0=sumi0
      xi1=sumi1
      xi2=sumi2
      xi3=sumi3
      write(9) uu0(1),uu0(3),c,ugr,xi0,xi1,xi2,xi3,are,flagr
      do 500 i=1,mmax
      write(9) depth(i),ur(i),uz(i),tz(i),tr(i),
     *             dcda(i),dcdb(i),dcdr(i)
  500 continue
  510 continue
      do 560 i=1,ks
      j=nos(i)
      urs=ur(j)
      uzs=uz(j)
      durs=-wvno*uzs+tr(j)/xmu(j)
      duzs=(wvno*xlam(j)*urs+tz(j))/(xlam(j)+2.*xmu(j))
      ur0=ur(11)
c
c     output
c
      write(3) wvno,ur0,are,ugr,gamma
      write(3) urs,durs,uzs,duzs
  560 continue
  600 continue
      go to 400
  700 continue
c
c     output the eigenfunction files for different source depths.
c
      if(kks.le.0) go to 950
      do 900 i=1,ks
      rewind 3
      write(6,*)
     * 'enter the name of output file storing the eigenfunctions'
      write(6,60) dphs(i)
      read(5,5) names
      open(4,file=names,status='new',form='unformatted')
      rewind 4
      write(4)
     *   mmax,(d(j),a(j),b(j),rho(j),qa1(j),qb1(j),j=1,mmax)
      write(4) nper,dphs(i)
  800 continue
      read(3) ifunc,mode,t
      write(4) ifunc,mode,t
      if(ifunc.lt.0) go to 870
      if(mode.le.0) go to 800
      do 860 k=1,mode
      do 850 j=1,ks
```

```
      read(3) qa(j),qb(j),dphq(j),depth(j),dcda(j)
      read(3) ur(j),tr(j),uz(j),tz(j)
 850  continue
      write(4) qa(i),qb(i),dphq(i),depth(i),dcda(i)
      write(4) ur(i),tr(i),uz(i),tz(i)
 860  continue
      go to 800
 870  continue
      close(4)
 900  continue
 950  continue
      close(1)
      close(2)
      close(3,status='delete')
      write(6,*) ' '
      write(6,*) 'reigen81 finished'
      write(6,*) ' '
      stop
      end
c
c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine gammap(omega,wvno,gamma)
c     This routine finds the attenuation gamma value of
c     surface wave.
c
      common/model/  d(100),a(100),b(100),rho(100),qa1(100),
     *                 qb1(100),xmu(100),xlam(100),mmax,ll
      common/eigfun/ ur(100),uz(100),tz(100),tr(100),uu0(4),
     *                 dcda(100),dcdb(100),dcdr(100)
      x=0.0
      do 100 i=ll,mmax
      x=x+dcda(i)*a(i)/qa1(i)+dcdb(i)*b(i)/qb1(i)
 100  continue
      c=omega/wvno
      gamma=0.5*wvno*x/c
      return
      end
c
c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine svfunc(omega,wvno)
c     This routine combines the Haskell vector from sub down and
c     Dunkin vector from sub up to form the eigenfunctions.
c
      double precision exe(100),exa(100),ext,fact
      common/model/  d(100),a(100),b(100),rho(100),qa1(100),
     *                 qb1(100),xmu(100),xlam(100),mmax,ll
      common/eigfun/ ur(100),uz(100),tz(100),tr(100),uu0(4),
     *                 dcda(100),dcdb(100),dcdr(100)
      common/dunk/   uu(100,5),exe,exa
      common/hask/   vv(100,4)
      common/water/ water0,ktrig
```

```
      ktrig=1
      call up(omega,wvno,fr)
      ktrig=0
      call down(omega,wvno)
      wvno2=wvno*wvno
      omega2=omega*omega
      f5=uu(ll,4)
      uu0(1)=wvno*uu(ll,3)/f5
      uu0(2)=1.0
c     uu0(3)=tz is actually the period equation.
      uu0(3)=fr
c     uu0(4)=tr should be zero.
      uu0(4)=fr
      ur(ll)=uu(ll,3)/f5
      uz(ll)=1.0
      tz(ll)=-water0
      tr(ll)=0.0
      do 200 i=ll+1,mmax
      i1=i-1
      uu1 =
     *  vv(i1,2)*uu(i,1)+vv(i1,3)*uu(i,2)      +vv(i1,4)*uu(i,3)
      uu2 =
     * -vv(i1,1)*uu(i,1)-vv(i1,3)*uu(i,3)*wvno2+vv(i1,4)*uu(i,4)
      uu3 =
     * -vv(i1,1)*uu(i,2)+vv(i1,2)*uu(i,3)*wvno2+vv(i1,4)*uu(i,5)
      uu4 =
     * -vv(i1,1)*uu(i,3)-vv(i1,2)*uu(i,4)      -vv(i1,3)*uu(i,5)
      ext=0.0
      do 100 k=ll,i1
      ext=ext+exa(k)-exe(k)
  100 continue
      fact=0.0
      if(ext.gt.-80.0) fact=dexp(ext)
      ur(i)=uu1*fact/f5
      uz(i)=uu2*fact/f5
      tz(i)=uu3*fact/f5
      tr(i)=uu4*fact/f5
  200 continue
      return
      end
c
c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine up(omega,wvno,fr)
c     This routine finds the values of the Dunkin vectors at
c       each layer boundaries from bottom layer upward.
c
      dimension ee0(5)
      double precision exe(100),exa(100),ex1,ex2,p,q,rab,dept
      double precision wd,wra,wcosd2,wsin2d
      common/model/  d(100),a(100),b(100),rho(100),qa1(100),
     *                 qb1(100),xmu(100),xlam(100),mmax,ll
      common/dunk/   uu(100,5),exe,exa
```

```
      common/save/    dd(5,5),aa(4,4),ex1,ex2
      common/aamatx/ ww(100),xx(100),yy(100),zz(100),
     *                cospp(100),cosqq(100)
      common/water/ water0,ktrig
      common/engerw/ wd,wra,wcosd2,wsin2d
      wvno2=wvno*wvno
      xka=omega/a(mmax)
      xkb=omega/b(mmax)
      ra=sqrt(abs(wvno2-xka*xka))
      rb=sqrt(abs(wvno2-xkb*xkb))
      t = b(mmax)/omega
      gammk = 2.*t*t
      gam = gammk*wvno2
      gamm1 = gam - 1.
      if(wvno.lt.xka) write(6,*) ' imaginary nua'
      if(wvno.lt.xkb) write(6,*) ' imaginary nub'
      uu(mmax,1)=wvno2-ra*rb
      uu(mmax,2)=-rho(mmax)*rb
      uu(mmax,3)=rho(mmax)*(gamm1-gammk*ra*rb)
      uu(mmax,4)=rho(mmax)*ra
      uu(mmax,5)=rho(mmax)*rho(mmax)*(gamm1*gamm1-gam*gammk*ra*rb)
c     matrix multiplication from bottom layer upward
      mmx1=mmax-1
      do 400 k=mmx1,11,-1
      k1=k+1
      dpth=d(k)
      xka = omega/a(k)
      xkb = omega/b(k)
      t = b(k)/omega
      gammk = 2.*t*t
      gam = gammk*wvno2
      ra=abs(wvno2-xka*xka)
      rab=dble(ra)
      rab=dsqrt(rab)
      dept=dble(dpth)
      p=rab*dept
      ra=sngl(rab)
      rb=abs(wvno2-xkb*xkb)
      rab=dble(rb)
      rab=dsqrt(rab)
      q=rab*dept
      rb=sngl(rab)
      call var(k,p,q,ra,rb,wvno,xka,xkb,dpth)
      call dnka(wvno2,gam,gammk,rho(k))
      exa(k)=ex2
      do 200 i=1,5
      cc=0.0
      do 100 j=1,5
      cc=cc+dd(i,j)*uu(k1,j)
  100 continue
      ee0(i)=cc
  200 continue
      call normc(ee0,rab)
```

```
      exe(k)=ex1+rab
      do 300 i=1,5
      uu(k,i)=ee0(i)
  300 continue
  400 continue
      water0=0.0
      if(ll.eq.1) go to 500
      xka=omega/a(1)
      ra=abs(wvno2-xka*xka)
      rab=dble(ra)
      rab=dsqrt(rab)
      wra=rab
      dept=dble(d(1))
      p=rab*dept
      ra=sngl(rab)
      call var(100,p,0.0,ra,0.0,wvno,xka,0.0,d(1))
c     water0 describes the surface water layer effect.
      water0=rho(1)*ww(100)/cospp(100)
      if(ktrig.eq.0) go to 500
c     prepare for subroutine wenerg which takes the energy
c     integral over water layer.
      q=2.*ex2
      rab=0.0
      if(q.gt.-80.0.and.q.lt.80.0) rab=1./dexp(q)
      wd=rab*dept
      wcosd2=2.*cospp(100)*cospp(100)
      p=2.*p
      ra=2.*wra
      call var(100,p,0.0,ra,0.0,wvno,xka,0.0,d(1))
      q=ex2-q
      rab=0.0
      if(q.gt.-80.) rab=dexp(q)
      wsin2d=rab*ww(100)
      if(wvno.lt.xka) wra=-wra
  500 continue
      fr=uu(ll,5)+water0*uu(ll,4)
      return
      end
c
c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine down(omega,wvno)
c     This routine finds the values of the Haskell vectors at
c     each layer boundaries from top layer downward.
c
      dimension aa0(5)
      double precision exe(100),exa(100),ex1,ex2
      common/model/  d(100),a(100),b(100),rho(100),qa1(100),
     *                 qb1(100),xmu(100),xlam(100),mmax,ll
      common/dunk/   uu(100,5),exe,exa
      common/hask/   vv(100,4)
      common/save/   dd(5,5),aa(4,4),ex1,ex2
      common/aamatx/ ww(100),xx(100),yy(100),zz(100),
```

```
     *                    cospp(100),cosqq(100)
      wvno2=wvno*wvno
      do 100 j=1,4
      vv(11,j)=0.0
 100  continue
      vv(11,4)=1.0
      aa0(5)=0.0
      mmx1=mmax-1
      do 500 k=11,mmx1
      k1=k-1
      if(k.eq.11) k1=11
      t=b(k)/omega
      gammk=2.*t*t
      gam=gammk*wvno2
      w=ww(k)
      x=xx(k)
      y=yy(k)
      z=zz(k)
      cosp=cospp(k)
      cosq=cosqq(k)
      call hska(w,x,y,z,cosp,cosq,wvno2,gam,gammk,rho(k))
      do 300 j=1,4
      cc=0.0
      do 200 i=1,4
      cc=cc+vv(k1,i)*aa(i,j)
 200  continue
      aa0(j)=cc
 300  continue
      call normc(aa0,ex2)
      exa(k)=exa(k)+ex2
      do 400 i=1,4
      vv(k,i)=aa0(i)
 400  continue
 500  continue
      return
      end
c
c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine var(m,pp,qq,ra,rb,wvno,xka,xkb,dpth)
c     This routine calculates the values of exp(p),exp(q)...
c     Since exp(88) ~ 10.**38, the exponential power terms p,q
c     must be controlled to prevent possible overflow.
c
      double precision exa,ex,pp,qq,qpp,qmp
      double precision expp,expm,exqp,exqm,sinp,cosp,sinq,cosq
      common/save/   dd(5,5),aa(4,4),exa,ex
      common/aamatx/ w0(100),x0(100),y0(100),z0(100),
     *                cosp0(100),cosq0(100)
      common/ovrflw/ a0,cpcq,cpy,cpz,cqw,cqx,xy,xz,wy,wz
      ex=0.0
      a0=1.0
      if(wvno-xka) 100,200,300
```

```
100 sinp=dsin(pp)
    w=sinp/dble(ra)
    x=-dble(ra)*sinp
    cosp=dcos(pp)
    go to 500
200 cosp=1.0d+0
    w=dpth
    x=0.0
    go to 500
300 if(pp.gt.40.0) go to 400
    expp=dexp(pp)
    expm=1./expp
    sinp=(expp-expm)*0.5
    cosp=(expp+expm)*0.5
    w=sinp/dble(ra)
    x=dble(ra)*sinp
    go to 500
400 ex=pp
    x=ra*0.5
    w=0.5/ra
    cosp=0.5
    a0=0.0
    if(ex.lt.75.0) a0=1./dexp(ex)
500 continue
    if(m.eq.100) go to 1100
    if(wvno-xkb) 600,700,800
600 sinq=dsin(qq)
    z=-dble(rb)*sinq
    y=sinq/dble(rb)
    cosq=dcos(qq)
    go to 900
700 cosq=1.0d+0
    y=dpth
    z=0.0
    go to 900
800 if(qq.gt.40.0) go to 1000
    exqp=dexp(qq)
    exqm=1./exqp
    sinq=(exqp-exqm)*0.5
    cosq=(exqp+exqm)*0.5
    y=sinq/dble(rb)
    z=dble(rb)*sinq
900 cpcq=cosp*cosq
    cpy=cosp*y
    cpz=cosp*z
    cqw=cosq*w
    cqx=cosq*x
    xy=x*y
    xz=x*z
    wy=w*y
    wz=w*z
    cosq=a0*cosq
    z=a0*z
```

```
      y=a0*y
      exa=ex
      go to 1100
 1000 qpp=qq+pp
      qmp=qq-pp
      exqp=0.0
      if(abs(qmp).lt.60.) exqp=dexp(qmp)
      exqm=0.0
      if(qpp.lt.60.) exqm=1./dexp(qpp)
      sinq=(exqp-exqm)*0.5
      cosq=(exqp+exqm)*0.5
      y=sinq/dble(rb)
      z=dble(rb)*sinq
      zz=rb*0.5
      yy=0.5/rb
      ccosq=0.5
      exa=ex+qq
      a0=0.0
      if(exa.lt.75.0) a0=1./dexp(exa)
      cpcq=cosp*ccosq
      cpy=cosp*yy
      cpz=cosp*zz
      xy=x*yy
      xz=x*zz
      wy=w*yy
      wz=w*zz
      cqw=ccosq*w
      cqx=ccosq*x
 1100 continue
      w0(m)=w
      x0(m)=x
      y0(m)=y
      z0(m)=z
      cosp0(m)=cosp
      cosq0(m)=cosq
      return
      end
c
c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine normc(ee,ex)
c     This routine is an important step to control over- or
c     underflow.
c     The Haskell or Dunkin vectors are normalized before
c     the layer matrix stacking.
c     Note that some precision will be lost during normalization.
c
      dimension ee(5)
      double precision ex,t1,t2
      t0 = 0.0
      do 10 i = 1,5
      if(abs(ee(i)).gt.t0) t0 = abs(ee(i))
   10 continue
```

```
      t1=dble(t0)
      if(t1.lt.1.d-30) t1=1.d+00
      do 20 i =1,5
      t2=dble(ee(i))
      t2=t2/t1
      ee(i)=sngl(t2)
   20 continue
      ex=dlog(t1)
      return
      end
c
c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine hska(w,x,y,z,cosp,cosq,wvno2,gam,gammk,rho)
      double precision ex1,ex2
      common/save/ dd(5,5),aa(4,4),ex1,ex2
      gamm1 = gam-1.
      temp = x-wvno2*y
      aa(2,3) = temp/rho
      temp = temp*gammk
      aa(4,3) = temp+y
      aa(4,1) = -(gamm1*y+gam*aa(4,3))*rho
      aa(2,1) = -wvno2*aa(4,3)
      temp = cosq-cosp
      aa(1,3) = temp/rho
      aa(2,4) = -wvno2*aa(1,3)
      aa(4,2) = rho*gammk*gamm1*temp
      aa(3,1) = -wvno2*aa(4,2)
      temp = temp*gam
      aa(1,1) = cosq-temp
      aa(2,2) = cosp+temp
      aa(3,3) = aa(2,2)
      aa(4,4) = aa(1,1)
      temp = z-wvno2*w
      aa(1,4) = temp/rho
      aa(1,2) = -w-gammk*temp
      aa(3,4) = -wvno2*aa(1,2)
      aa(3,2) = rho*(gam*aa(1,2)-gamm1*w)
      return
      end
c
c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine dnka(wvno2,gam,gammk,rho)
      double precision ex1,ex2
      common/save/ dd(5,5),aa(4,4),ex1,ex2
      common/ovrflw/ a0,cpcq,cpy,cpz,cqw,cqx,xy,xz,wy,wz
      gamm1 = gam-1.
      twgm1=gam+gamm1
      gmgmk=gam*gammk
      gmgm1=gam*gamm1
      gm1sq=gamm1*gamm1
      rho2=rho*rho
```

```
      aOpq=aO-cpcq
      dd(1,1)=cpcq-2.*gmgm1*aOpq-gmgmk*xz-wvno2*gm1sq*wy
      dd(2,1)=(gm1sq*cqw-gmgmk*cpz)*rho
      dd(3,1)=-(gammk*gamm1*twgm1*aOpq+gam*gammk*gammk*xz+
     *             gamm1*gm1sq*wy)*rho
      dd(4,1)=(gmgmk*cqx-gm1sq*cpy)*rho
      dd(5,1)=-(2.*gmgmk*gm1sq*aOpq+gmgmk*gmgmk*xz+
     *             gm1sq*gm1sq*wy)*rho2
      dd(1,2)=(cqx-wvno2*cpy)/rho
      dd(2,2)=cpcq
      dd(3,2)=gammk*cqx-gamm1*cpy
      dd(4,2)=-xy
      dd(5,2)=dd(4,1)
      dd(1,4)=(wvno2*cqw-cpz)/rho
      dd(2,4)=-wz
      dd(3,4)=gamm1*cqw-gammk*cpz
      dd(4,4)=dd(2,2)
      dd(5,4)=dd(2,1)
      dd(1,5)=-(2.*wvno2*aOpq+xz+wvno2*wvno2*wy)/rho2
      dd(2,5)=dd(1,4)
      dd(3,5)=-(twgm1*aOpq+gammk*xz+wvno2*gamm1*wy)/rho
      dd(4,5)=dd(1,2)
      dd(5,5)=dd(1,1)
      t=-2.*wvno2
      dd(1,3)=t*dd(3,5)
      dd(2,3)=t*dd(3,4)
      dd(3,3)=aO+2.*(cpcq-dd(1,1))
      dd(4,3)=t*dd(3,2)
      dd(5,3)=t*dd(3,1)
      return
      end
c
c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine energy(omega,wvno)
c     This routine takes the energy integral by an analytic
c     way using the eigenfuntions found above.
c
      double precision wvnoO,omegaO
      double precision wvno2,omega2,wvomg2,sum,xka,xkb,ra,rb
      double precision dpth,daa,dbb,drho,dlam,dlamu,dmu
      double precision urur,urtz,uzuz,uztr,trtr,urduz,uzdur
      double precision durdur,duzduz,dldl,dldm,dldk,dldr
      double precision sumiO,sumi1,sumi2,sumi3
      double complex t(4,4),tt(4,4),ff(6),pp(4)
      double complex nua,nub,c1,c2
      common/coef/   t,tt,ff,pp
      common/model/  d(100),a(100),b(100),rho(100),qa1(100),
     *                 qb1(100),xmu(100),xlam(100),mmax,ll
      common/eigfun/ uu(100,4),uuO(4),dcda(100),dcdb(100),
     *                 dcdr(100)
      common/sumi/   sumiO,sumi1,sumi2,sumi3,flagr,are,ugr
      wvnoO=dble(wvno)
```

```
      omega0=dble(omega)
      omega2=omega0*omega0
      wvomg2=wvno0*omega2
      wvno2=wvno0*wvno0
      sumi0=0.0d+00
      sumi1=0.0d+00
      sumi2=0.0d+00
      sumi3=0.0d+00
      do 300 k=11,mmax
      k1=k+1
      kk=k
      if(k.eq.mmax) kk=101
      daa=dble(a(k))
      dbb=dble(b(k))
      drho=dble(rho(k))
      dlam=dble(xlam(k))
      dmu=dble(xmu(k))
      dlamu=dlam+2.*dmu
      dpth=dble(d(k))
      xka=omega0/daa
      xkb=omega0/dbb
      gammk=dbb/omega0
      gammk=2.*gammk*gammk
      gam=gammk*wvno2
      ra=dsqrt(dabs(wvno2-xka*xka))
      rb=dsqrt(dabs(wvno2-xkb*xkb))
      if(ra.lt.1.d-6) ra=1.d-6
      if(rb.lt.1.d-6) rb=1.d-6
      nua=dcmplx(ra,0.0)
      nub=dcmplx(rb,0.0)
      if(wvno0.lt.xka) nua=dcmplx(0.0,ra)
      if(wvno0.lt.xkb) nub=dcmplx(0.0,rb)
      call tminus(nua,nub,wvno2,gam,gammk,drho)
      call tplus(nua,nub,wvno2,gam,gammk,drho)
      call intval(kk,nua,nub,dpth)
      do 200 i=1,2
      i2=i+2
      c1=0.0
      c2=0.0
      do 100 j=1,4
      c1=c1+tt(i,j)*uu(k1,j)
      c2=c2+tt(i2,j)*uu(k,j)
100   continue
      pp(i)=c1
      pp(i2)=c2
200   continue
      urur=sum(kk,1,1)*wvno2
      urtz=sum(kk,1,3)*wvomg2
      uzuz=sum(kk,2,2)
      uztr=sum(kk,2,4)*wvomg2
      tztz=sum(kk,3,3)*omega2*omega2
      trtr=sum(kk,4,4)*wvomg2*wvomg2
      urduz=(wvno0*dlam*urur+urtz)/dlamu
```

```
      uzdur=-wvno0*uzuz+uztr/dmu
      durdur=wvno2*uzuz-2.*wvno0*uztr/dmu+trtr/(dmu*dmu)
      duzduz=(wvno2*dlam*dlam*urur+2.*wvno0*dlam*urtz+
     *            tztz)/(dlamu*dlamu)
      sumi0=sumi0+drho*(uzuz+urur)
      sumi1=sumi1+dlamu*urur+dmu*uzuz
      sumi2=sumi2+dmu*uzdur-dlam*urduz
      sumi3=sumi3+dlamu*duzduz+dmu*durdur
      dldl=-wvno2*urur+2.*wvno0*urduz-duzduz
      dldm=-wvno2*(2.*urur+uzuz)-2.*wvno0*uzdur-(2.*duzduz+durdur)
      dldr=omega2*(urur+uzuz)
      dcda(k)=2.*drho*daa*omega0*dldl/wvno2
      dcdb(k)=2.*drho*dbb*omega0*(dldm-2.*dldl)/wvno2
      dcdr(k)=dldr+dlam*dldl/drho+dmu*dldm/drho
      dcdr(k)=dcdr(k)*omega0/wvno2
 300  continue
      if(b(1).le.0.0) call wenerg(wvno0)
      dldk=-2.*(wvno0*sumi1+sumi2)
      do 400 k=11,mmax
      dcda(k)=dcda(k)/dldk
      dcdb(k)=dcdb(k)/dldk
      dcdr(k)=dcdr(k)/dldk
 400  continue
      flagr=omega2*sumi0-wvno2*sumi1-2.*wvno*sumi2-sumi3
      ugr=(wvno0*sumi1+sumi2)/(omega0*sumi0)
      are=wvno0/(2.*omega0*ugr*sumi0)
      return
      end
c
c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      function sum(kk,i,j)
c
c     The analytic forms of the solution of integral:
c     Integral U*U dz = T-matrix * eigenfnction * integral-coefs
c
      double precision sum
      double complex t(4,4),tt(4,4),ff(6),pp(4)
      double complex sum0,sum1,sum2,sum3,sum4,sum5,sum6
      common/coef/ t,tt,ff,pp
      if(kk.eq.101) go to 100
      sum1=t(i,1)*t(j,1)*pp(1)*pp(1)+t(i,3)*t(j,3)*pp(3)*pp(3)
      sum1=sum1*ff(1)
      sum2=t(i,2)*t(j,2)*pp(2)*pp(2)+t(i,4)*t(j,4)*pp(4)*pp(4)
      sum2=sum2*ff(2)
      sum3=(t(i,1)*t(j,2)+t(i,2)*t(j,1))*pp(1)*pp(2)
     *        +(t(i,3)*t(j,4)+t(i,4)*t(j,3))*pp(3)*pp(4)
      sum3=sum3*ff(3)
      sum4=(t(i,1)*t(j,4)+t(i,4)*t(j,1))*pp(1)*pp(4)
     *        +(t(i,3)*t(j,2)+t(i,2)*t(j,3))*pp(2)*pp(3)
      sum4=sum4*ff(4)
      sum5=(t(i,1)*t(j,3)+t(i,3)*t(j,1))*pp(1)*pp(3)
      sum5=sum5*ff(5)
```

```
      sum6=(t(i,2)*t(j,4)+t(i,4)*t(j,2))*pp(2)*pp(4)
      sum6=sum6*ff(6)
      sum0=sum1+sum2+sum3+sum4+sum5+sum6
      sum=real(sum0)
      return
 100  continue
      sum1=t(i,3)*t(j,3)*pp(3)*pp(3)*ff(1)
      sum2=t(i,4)*t(j,4)*pp(4)*pp(4)*ff(2)
      sum3=(t(i,3)*t(j,4)+t(i,4)*t(j,3))*pp(3)*pp(4)
      sum3=sum3*ff(3)
      sum0=sum1+sum2+sum3
      sum=real(sum0)
      return
      end
c
c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine intval(k,nua,nub,dpth)
c
c     This routine finds the coeficients needed for integrals.
c
      double precision dpth
      double complex t(4,4),tt(4,4),ff(6),pp(4)
      double complex nua,nub,p,q,pq,expp,exqq
      common/coef/ t,tt,ff,pp
      if(k.eq.101) go to 100
      p=nua*dpth
      q=nub*dpth
      pq=(nua+nub)*dpth
      call ifpq(p,p+p,expp)
      ff(1)=(1.0-expp)/(2.*nua)
      call ifpq(q,q+q,exqq)
      ff(2)=(1.0-exqq)/(2.*nub)
      call ifpq(pq/2 ,pq,expp)
      ff(3)=(1.0-expp)/(nua+nub)
      call ifpq(p/2.,p,expp)
      call ifpq(q/2.,q,exqq)
      ff(4)=(exqq-expp)/(nua-nub)
      ff(5)=dpth*expp
      ff(6)=dpth*exqq
      return
 100  continue
      ff(1)=0.5/nua
      ff(2)=0.5/nub
      ff(3)=1./(nua+nub)
      return
      end
c
c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine ifpq(p,pq,expq)
      double complex p,pq,expq
      if(real(p).lt.40.0) go to 100
```

```
      expq=0.0d+00
      go to 200
 100  continue
      expq=zexp(-pq)
 200  continue
      return
      end
c
c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine tplus(nua,nub,wvno2,gam,gammk,rho)
c
c     T matrix.
c
      double precision gam,gammk,rho,wvno2
      double complex t(4,4),tt(4,4),ff(6),pp(4)
      double complex nua,nub
      common/coef/ t,tt,ff,pp
      t(1,1)=-1./rho
      t(1,2)=nub/rho
      t(1,3)=t(1,1)
      t(1,4)=-t(1,2)
      t(2,1)=-nua/rho
      t(2,2)=wvno2/rho
      t(2,3)=-t(2,1)
      t(2,4)=t(2,2)
      t(3,1)=1.-gam
      t(3,2)=gam*nub
      t(3,3)=t(3,1)
      t(3,4)=-t(3,2)
      t(4,1)=-gammk*nua
      t(4,2)=-t(3,1)
      t(4,3)=-t(4,1)
      t(4,4)=t(4,2)
      do 100 i=1,4
      do 100 j=1,4
      t(i,j)=0.5*t(i,j)
 100  continue
      return
      end
c
c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine tminus(nua,nub,wvno2,gam,gammk,rho)
c
c     T-inverse matrix
c
      double precision gam,gamm1,gammk,rho,wvno2
      double complex t(4,4),tt(4,4),ff(6),pp(4)
      double complex nua,nub
      common/coef/ t,tt,ff,pp
      gamm1=gam-1.0
      tt(1,1)=-rho*gam
```

```
      tt(1,2)=rho*gamm1/nua
      tt(1,3)=1.0
      tt(1,4)=-wvno2/nua
      tt(2,1)=-rho*gamm1/nub
      tt(2,2)=rho*gammk
      tt(2,3)=1./nub
      tt(2,4)=-1.0
      tt(3,1)=tt(1,1)
      tt(3,2)=-tt(1,2)
      tt(3,3)=1.0
      tt(3,4)=-tt(1,4)
      tt(4,1)=-tt(2,1)
      tt(4,2)=tt(2,2)
      tt(4,3)=-tt(2,3)
      tt(4,4)=-1.0
      return
      end
c
c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine wenerg(wvno)
c     calculate energy trapped in the top water layer.
      double precision wvno,wd,wra,wra2,wcosd2,wsin2d
      double precision urur,uzuz,urduz,duzduz,wvno2,dr,dlam
      double precision sumi0,sumi1,sumi2,sumi3
      common/model/  d(100),a(100),b(100),rho(100),qai(100),
     *               qbi(100),xmu(100),xlam(100),mmax,ll
      common/sumi/   sumi0,sumi1,sumi2,sumi3,flagr,are,ugr
      common/engerw/ wd,wra,wcosd2,wsin2d
      wvno2=wvno*wvno
      wra2=wra*dabs(wra)
      urur  =(wsin2d-wd)/(wra2*wcosd2)
      urur  =urur*wvno2
      uzuz  =(wsin2d+wd)/wcosd2
      urduz =(wsin2d-wd)/wcosd2
      urduz =urduz*wvno
      duzduz=(wsin2d-wd)*wra2/wcosd2
      dr=dble(rho(1))
      dlam=dble(xlam(1))
      sumi0=sumi0+dr*(urur+uzuz)
      sumi1=sumi1+dlam*urur
      sumi2=sumi2-dlam*urduz
      sumi3=sumi3+dlam*duzduz
      return
      end
*********
```

```fortran
c       leigen81.f
c
c       f77 -i -12 leigen81.f -o leigen81
c
c       This is a new version of program leigen which calculates
c       the eigenfunctions of Love wave for any plane layered
c       model.
c
c       Some analytic ofms for energy integrals are used instead
c       of taking numerical integration directly.
c
c       The I/O are mostly the same as those of program leigen,
c       except that the eigenfunctions calculated are located at
c       the top of each layer, not in the middle.
c
c       The Q model is taken into account.
c
c       For the sake of space saving, only the values of
c       eigenfunction at the source depth are stored.
c       However, several files with different source depths
c       can be set up.
c
c       -Oct 10, 1981
        dimension nos(100),dphs(100),dphq(100),qa(100),qb(100)
        dimension depth(100)
        common/model/  d(100),a(100),b(100),rho(100),qa1(100),
     *                 qb1(100),xmu(100),xlam(100),mmax,ll
        common/eigfun/ ut(100),tt(100),dcdb(100),dcdr(100),uu0(4)
        common/sumi/   sumi0,sumi1,sumi2,flagr,ale,ugr
        character*1 dd
        character*50 names
  5     format(a)
 10     format(/2x,'M',3x,'DPTH',3x,'  D  ',3x,'  A  ',3x,'  B  ',
     *         3x,' RHO',3x,' QA  ',4x,' QB  ',4x,' MU  ',3x,'LMDA')
 20     format(i3,1x,5f7.2,2f8.2,2f7.2)
 30     format(3x,'-source is at the top of this layer.',
     *         ' source depth = ',f6.2)
 40     format(i3,1x,f7.2,7x,3f7.2,2f8.2,2f7.2)
 50     format(3x,'-source is inside the half space.',
     *         'source depth = ',f6.2)
 60     format(' at the source depth = ',f6.2,' km')
        write(6,*) 'enter the input file name: (from surface81)'
        read(5,5) names
        open(1,file=names,status='old',form='unformatted')
        rewind 1
        open(3,file='tmp.1',status='scratch',form='unformatted')
        rewind 3
c
c       enter the source depths and Q model.
c
        write(6,*) 'enter the source depths:'
        write(6,*)
     *   '(no. of source depths, srcdph(1),srcdph(2),...)'
```

```
      write(6,*)
     *    '*** if no. of source depths is negative, no output'
      write(6,*) '     files will be generated.***'
      read(5,*) kks,(dphs(i),i=1,iabs(kks))
      ks=iabs(kks)
      write(6,*) 'enter Q-model here(1), or from file(2) or'
      write(6,*) 'NOT consider Q in this program(3):'
      read(5,*) kk
      kq=1
      go to (100,110,170),kk
 100  continue
      write(6,*)
     *    'enter d(i),Qa(i),Qb(i)    use d(i)=0 for halfspace'
      icode=5
      go to 120
 110  continue
      write(6,*) 'enter the name of the file storing Q-model:'
      read(5,5) names
      open(2,file=names,status='old',form='formatted')
      rewind 2
      icode=2
 120  continue
      i=1
      base=0.0
 140  continue
      read(icode,*) dq0,qa(i),qb(i)
      if(dq0.le.0.0) go to 150
      base=base+dq0
      dphq(i)=base
      i=i+1
      go to 140
 150  kq=i
      do 160 i=1,100
      qa1(i)=qa(kq)
      qb1(i)=qb(kq)
 160  continue
 170  continue
      write(6,*) 'Store the derivatives? (y/n)'
      read(5,5) dd
      if(dd.eq.'n') go to 175
      write(6,*) 'enter the output file name for derivatives:'
      read(5,5) names
      open(9,file=names,status='new',form='unformatted')
      rewind 9
 175  continue
      do 180 i=1,100
      depth(i)=1000000.0
 180  continue
c
c     enter the earth model.
c
      read(1) mmax
      write(6,10)
```

```
      base=0.0
      depth(1)=0.0
      do 190 i=1,mmax
      read(1) d(i),a(i),b(i),rho(i)
      base=base+d(i)
      depth(i+1)=base
      xmu(i)=rho(i)*b(i)*b(i)
      xlam(i)=rho(i)*(a(i)*a(i)-2.*b(i)*b(i))
  190 continue
c
c     insert the Q-model into the velocity model.
c     insert the source depth at the boundary of a layer.
c
      kqs=kq+ks-1
      do 300 k0=1,kqs
      dph0=dphq(k0)
      is=k0-kq+1
      if(k0.ge.kq) dph0=dphs(is)
      do 200 i=1,mmax
      k=i
      if(dph0.eq.depth(i)) go to 250
      if(dph0.gt.depth(i).and.dph0.lt.depth(i+1)) go to 210
  200 continue
  210 k1=k+1
      dphk1=depth(k1)
      do 220 i=mmax,k,-1
      i1=i+1
      d(i1)=d(i)
      a(i1)=a(i)
      b(i1)=b(i)
      rho(i1)=rho(i)
      xmu(i1)=xmu(i)
      xlam(i1)=xlam(i)
      depth(i1)=depth(i)
      if(k0.ge.kq) qa1(i1)=qa1(i)
      if(k0.ge.kq) qb1(i1)=qb1(i)
  220 continue
      d(k)=dph0-depth(k)
      d(k1)=dphk1-dph0
      depth(k1)=depth(k)+d(k)
      mmax=mmax+1
      if(k0.ge.kq) go to 240
      if(k0.eq.1) ns=1
      do 230 j=ns,k
      qa1(j)=qa(k0)
      qb1(j)=qb(k0)
  230 continue
  240 ns=k1
      go to 280
  250 continue
      if(k0.ge.kq) go to 270
      if(k0.eq.1) ns=1
      do 260 i=ns,k-1
```

```
      qa1(i)=qa(k0)
      qb1(i)=qb(k0)
  260 continue
  270 ns=k
  280 nos(is)=ns
  300 continue
      j=1
      do 310 i=1,mmax-1
      write(6,20) i,depth(i),d(i),a(i),b(i),rho(i),qa1(i),qb1(i),
     *                 xmu(i),xlam(i)
      if(i.ne.nos(j)) go to 310
      write(6,30) dphs(j)
      j=j+1
  310 continue
      i=mmax
      write(6,40) i,depth(i),
     *      a(i),b(i),rho(i),qa1(i),qb1(i),xmu(i),xlam(i)
      if(nos(j).eq.mmax) write(6,50) dphs(j)
      if(dd.eq.'y')
     *   write(9) mmax,(d(i),a(i),b(i),rho(i),i=1,mmax)
      write(6,*) ' '
      write(6,*) 'wait.'
      ll=1
      if(b(1).le.0.0) ll=2
      read(1) nper
  400 continue
c
c     read in the dispersion values.
c
      read(1) ifunc,mode,t
      write(3) ifunc,mode,t
      if(dd.eq.'y') write(9) ifunc,mode,t
      if(ifunc.lt.0) go to 700
      if(mode.le.0) go to 400
      do 600 k=1,mode
      read(1) c
c
c     main part.
c
      omega=6.2831853/t
      wvno=omega/c
      call shfunc(omega,wvno)
      call energy(omega,wvno)
      if(kk.ne.3) call gammaq(omega,wvno,gamma)
      if(dd.eq.'n') go to 510
c     output the derivatives.
      write(9) dum,uu0(2),c,ugr,sumi0,sumi1,sumi2,dum,ale,flagr
      do 500 i=1,mmax
      write(9) depth(i),ut(i),tt(i),dum,dum,dum,dcdb(i),dcdr(i)
  500 continue
  510 continue
      do 560 i=1,ks
      j=nos(i)
```

```
      uts=ut(j)
      duts=tt(j)/xmu(j)
      write(3) wvno,ale,ugr,gamma,uts,duts
 560  continue
 600  continue
      go to 400
 700  continue
c
c     output the data files with different source depths.
c
      if(kks.le.0) go to 950
      do 900 i=1,ks
      rewind 3
      write(6,*)
     * 'enter the name of output file storing the eigenfunctions'
      write(6,60) dphs(i)
      read(5,5) names
      open(4,file=names,status='new',form='unformatted')
      rewind 4
      write(4)
     *   mmax,(d(j),a(j),b(j),rho(j),qa1(j),qb1(j),j=1,mmax)
      write(4) nper,dphs(i)
 800  continue
      read(3) ifunc,mode,t
      write(4) ifunc,mode,t
      if(ifunc.lt.0) go to 870
      if(mode.le.0) go to 800
      do 860 k=1,mode
      do 850 j=1,ks
      read(3) qa(j),qb(j),dphq(j),ut(j),tt(j),dcdb(j)
 850  continue
      write(4) qa(i),dummy,qb(i),dphq(i),ut(i)
      write(4) tt(i),dcdb(i),dummy,dummy
 860  continue
      go to 800
 870  continue
      close(4)
 900  continue
 950  continue
      close(1)
      close(2)
      close(3,status='delete')
      write(6,*) ' '
      write(6,*) 'leigen81 finished'
      write(6,*) ' '
      stop
      end
c
c  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine gammaq(omega,wvno,gamma)
c     This routine finds the attenuation gamma value.
c
```

```
      common/model/  d(100),a(100),b(100),rho(100),qa1(100),
     *                qb1(100),xmu(100),xlam(100),mmax,ll
      common/eigfun/ ut(100),tt(100),dcdb(100),dcdr(100),uu0(4)
      x=0.0
      do 100 i=ll,mmax
      x=x+dcdb(i)*b(i)/qb1(i)
 100  continue
      c=omega/wvno
      gamma=0.5*wvno*x/c
      return
      end
c
c  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine shfunc(omega,wvno)
c     This routine evaluates the eigenfunctions by calling sub
c     up.
c
      double precision exl(100),ext,fact
      common/model/  d(100),a(100),b(100),rho(100),qa1(100),
     *                qb1(100),xmu(100),xlam(100),mmax,ll
      common/eigfun/ uu(100,2),dcdb(100),dcdr(100),uu0(4)
      common/save/   exl
      call up(omega,wvno,f1)
      uu0(1)=1.0
c     uu0(2)=stress0 is actually the value of period equation.
c     uu0(3) is used to print out the period euation value before
c     the root is refined.
c
      uu0(2)=f1
      uu0(3)=0.0
      uu0(4)=0.0
      ext=0.0
      do 100 k=ll+1,mmax
      ext=ext+exl(k-1)
      fact=0.0
      if(ext.lt.85.0) fact=1./dexp(ext)
      uu(k,1)=uu(k,1)*fact/uu(ll,1)
      uu(k,2)=uu(k,2)*fact/uu(ll,1)
 100  continue
      uu(ll,1)=1.0
      uu(ll,2)=0.0
      return
      end
c
c  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine up(omega,wvno,f1)
c     This routine calculates the elements of Haskell matrix,
c     and finds the eigenfunctions by analytic solution.
c
      double precision exl(100),qq,rr,ss,exqm,exqp,sinq,cosq
      common/model/  d(100),a(100),b(100),rho(100),qa1(100),
```

```
     *                    qb1(100),xmu(100),xlam(100),mmax,ll
      common/eigfun/ uu(100,2),dcdb(100),dcdr(100),uu0(4)
      common/save/    exl
      wvno2=wvno*wvno
      xkb=omega/b(mmax)
      rb=sqrt(abs(wvno2-xkb*xkb))
      if(wvno.lt.xkb) write(6,*) ' imaginary nub'
      uu(mmax,1)=1.0
      uu(mmax,2)=-xmu(mmax)*rb
      mmx1=mmax-1
      do 500 k=mmx1,ll,-1
      k1=k+1
      dpth=d(k)
      xkb=omega/b(k)
      rb=abs(wvno2-xkb*xkb)
      rr=dble(rb)
      rr=dsqrt(rr)
      ss=dble(dpth)
      qq=rr*ss
      if(wvno-xkb) 100,200,300
100   sinq=dsin(qq)
      cosq=dcos(qq)
      y=sinq/rr
      z=-rr*sinq
      qq=0.0
      go  to 400
200   qq=0.0
      cosq=1.0d+0
      y=dpth
      z=0.0
      go to 400
300   if(qq.gt.40.0) go to 350
      exqp=1.
      exqm=1./dexp(qq+qq)
      sinq=(exqp-exqm)*0.5
      cosq=(exqp+exqm)*0.5
      y=sinq/rr
      z=rr*sinq
      go to 400
350   continue
      y=0.5/rr
      z=0.5*rr
      cosq=0.5
400   continue
      amp0=cosq*uu(k1,1)-y*uu(k1,2)/xmu(k)
      str0=cosq*uu(k1,2)-z*xmu(k)*uu(k1,1)
      rr=abs(amp0)
      ss=abs(str0)
      if(ss.gt.rr) rr=ss
      if(rr.lt.1.d-30) rr=1.d+00
      exl(k)=dlog(rr)+qq
      uu(k,1)=amp0/rr
      uu(k,2)=str0/rr
```

```
 500  continue
      f1=uu(11,2)
      return
      end
c
c  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine energy(omega,wvno)
c     This routine calculates the values of integrals I0, I1,
c     and I2 using analytic solutions. It is found
c     that such a formulation is more efficient and practical.
c
      double precision wvno0,omega0,c,sumi0,sumi1,sumi2
      double precision xkb,rb,dbb,drho,dpth,dmu,wvno2,omega2
      double precision upup,dupdup,dcb,dcr
      double complex nub,xnub,exqq,top,bot,f1,f2,f3
      common/model/  d(100),a(100),b(100),rho(100),qa1(100),
     *                  qb1(100),xmu(100),xlam(100),mmax,ll
      common/eigfun/ uu(100,2),dcdb(100),dcdr(100),uu0(4)
      common/sumi/   xi0,xi1,xi2,flagr,ale,ugr
      wvno0=dble(wvno)
      omega0=dble(omega)
      c=omega0/wvno0
      omega2=omega0*omega0
      wvno2=wvno0*wvno0
      sumi0=0.0d+00
      sumi1=0.0d+00
      sumi2=0.0d+00
      do 300 k=ll,mmax
      k1=k+1
      dbb=dble(b(k))
      drho=dble(rho(k))
      dpth=dble(d(k))
      dmu=dble(xmu(k))
      xkb=omega0/dbb
      rb=dsqrt(dabs(wvno2-xkb*xkb))
      if(k.eq.mmax) go to 100
      nub=dcmplx(rb,0.0)
      if(wvno0.lt.xkb) nub=dcmplx(0.0,rb)
      xnub=dmu*nub
      top=uu(k,1)-uu(k,2)/xnub
      bot=uu(k1,1)+uu(k1,2)/xnub
      f3=nub*dpth
      exqq=0.0
      if(real(f3).lt.40) exqq=zexp(-2.*f3)
      f1=(1.-exqq)/(2.*nub)
      exqq=0.0
      if(real(f3).lt.80) exqq=zexp(-f3)
      f2=dpth*exqq
      f1=0.25*f1*(top*top+bot*bot)
      f2=0.5 *f2*top*bot
      f3=f1+f2
      upup=real(f3)
```

```
      f3=xnub*xnub*(f1-f2)
      dupdup=real(f3)/(dmu*dmu)
      go to 200
 100  continue
      upup   =0.5/rb*uu(mmax,1)*uu(mmax,1)
      dupdup=0.5*rb*uu(mmax,1)*uu(mmax,1)
 200  continue
      sumi0=sumi0+drho*upup
      sumi1=sumi1+dmu*upup
      sumi2=sumi2+dmu*dupdup
      dcr=-0.5*c*c*c*upup
      dcb=0.5*c*(upup+dupdup/wvno2)
      dcdb(k)=2.*drho*dbb*dcb
      dcdr(k)=dcr+dbb*dbb*dcb
 300  continue
      do 400 k=11,mmax
      dcdb(k)=dcdb(k)/sumi1
      dcdr(k)=dcdr(k)/sumi1
 400  continue
      flagr=omega2*sumi0-wvno2*sumi1-sumi2
      ugr=sumi1/(c*sumi0)
      ale=0.5/sumi1
      xi0=sumi0
      xi1=sumi1
      xi2=sumi2
      return
      end
******
```

```
c       dpegn81.f
c
c       f77 -i -I2 dpegn81.f -o dpegn81 -lcalcomp12
c
c       plot dispersion curve
c       get data from reigen81 or leigen81
c
        dimension t(1000),v(1000),mode(500)
        dimension d(500),a(500),b(500),rho(500)
        character*20 names
        character*4 yorn,xlog,ylog
        common/ ctrl / yorn,xlog,ylog
 5      format(a)
 15     format(4f12.5)
 25     format(5x,i5,3x,e10.4,3x,e10.4)
        pi=2.*3.141592653
        write(6,*) ' '
        write(6,*) 'Rayleigh(1) or Love(2)?'
        read(5,*) llrr
        write(6,*) ' '
        if(llrr.eq.1) write(6,*)
     *    'enter input file name: (from reigen81)'
        if(llrr.eq.2) write(6,*)
     *    'enter input file name: (from leigen81)'
        read(5,5) names
        open(1,file=names,status='old',form='unformatted')
        write(6,*)
     *    'how many modes and which modes wanted? (e.g. 3,1,3,5)'
        write(6,*) '(if all modes wanted, answer  1,0 )'
        read(5,*) nmode,(mode(i),i=1,nmode)
        if(mode(1).ne.0) go to 40
        nmode=500
        do 30 i=1,500
        mode(i)=i
 30     continue
 40     continue
        write(6,*) 'plot U-perd(1)  U-freq(2)  C-perd(3)  C-freq(4)'
        write(6,*) '     K-freq(5)  freq-K(6)  K-C(7)     C-K(8)'
        write(6,*) '     Ar-perd(9)            Ar-freq(10)'
        write(6,*) '     gamma-perd(11)        gamma-freq(12)'
        write(6,*) '     Ar*atten-perd(13) Ar*atten-freq(14):'
        read(5,*) ipg
        write(6,*) 'plot X axis in log scale? (y/n):'
        read(5,5) xlog
        write(6,*) 'plot Y axis in log scale? (y/n):'
        read(5,5) ylog
        call plots(0,0,7)
        write(6,*) 'enter ipen:'
        read(5,*) ipen
        call newpen(ipen)
        yorn='y'
        call plot(2.5,2.3,-3)
        write(6,*) ' '
```

```
      write(6,*) 'model:'
      do 600 kk=1,nmode
      rewind 1
      read(1) nmax,(d(i),a(i),b(i),rho(i),ga1,gb1,i=1,nmax)
      if(kk.eq.1) write(6,15) (d(i),a(i),b(i),rho(i),i=1,nmax)
      read(1) nper,dphs
      kkmode=0
  100 continue
      ident=mode(kk)
      nt=0
      itrig=0
  200 continue
      read(1) ifunc,kmode,t1
      if(ifunc.lt.0) go to 400
      if(kmode.le.0) go to 200
      nt=nt+1
      do 300 i=1,kmode
      read(1) wvno0,ur0,are0,u0,gama0
      c0=pi/(t1*wvno0)
      read(1) ur0,dur0,uz0,duz0
      if(i.ne.ident) go to 300
      itrig=1
      nn=nt
      t(nn)=t1
      v(nn)=c0
      if(ipg.le.2) v(nn)=u0
      if(ipg.eq.9.or.ipg.eq.10) v(nn)=are0/sqrt(wvno0)
      if(ipg.eq.11.or.ipg.eq.12) v(nn)=gama0
      if(ipg.eq.13.or.ipg.eq.14)
     *   v(nn)=are0/(sqrt(wvno0)*exp(gama0*1000.0))
  300 continue
      go to 200
  400 continue
      if(itrig.eq.0.and.nmode.eq.500) go to 600
      if(t(nn).gt.5000.0) nn=nn-1
      do 550 i=1,nn
      go to (500,482,500,482,485,486,487,488,500,482,500,482,
     *         500,482),ipg
  482 t(i)=1./t(i)
      go to 500
  485 t(i)=1./t(i)
      v(i)=pi*t(i)/v(i)
      go to 500
  486 tmp=1./t(i)
      t(i)=pi*tmp/v(i)
      v(i)=tmp
      go to 500
  487 tmp=v(i)
      v(i)=pi/(t(i)*v(i))
      t(i)=tmp
      go to 500
  488 t(i)=pi/(t(i)*v(i))
  500 continue
```

```
      if(xlog.eq.'y') t(i)=log10(t(i))
      if(ylog.eq.'y') v(i)=log10(v(i))
  550 continue
      call disp(itrig,kk,ipg,ident,nn,t,v)
  600 continue
      call plot(0.0,8.0,999)
      close(1)
      close(2)
      write(6,*) ' '
      write(6,*) 'job finished'
      write(6,*) ' '
      stop
      end
c
      subroutine disp(itrig,kk,ipg,ident,nn,x,y)
      dimension x(1),y(1)
      character*4 yorn,nory,xlog,ylog
      character*11 alpha(2,14),alp,bet
      common/ ctrl / yorn,xlog,ylog
      data alpha/
     *  'U  (km/sec)','T  (sec)    ','U  (km/sec)','f  (Hz)     ',
     *  'C  (km/sec)','T  (sec)    ','C  (km/sec)','f  (Hz)     ',
     *  'K  (rad/km)','f  (Hz)     ','f  (Hz)    ','K  (rad/km)',
     *  'K  (rad/km)','C  (km/sec)','C  (km/sec)','K  (rad/km)',
     *  'Amp Factor ','T  (sec)    ','Amp Factor ','f  (Hz)     ',
     *  '   Gamma   ','T  (sec)    ','   Gamma   ','f  (Hz)     ',
     *  ' Ar*atten  ','T  (sec)    ',' Ar*atten  ','f  (Hz)     '/
      if(itrig.eq.0) write(6,*) 'mode ',ident,' is not generated.'
      if(itrig.eq.0) return
      alp=alpha(1,ipg)
      bet=alpha(2,ipg)
      if(yorn.eq.'n') go to 100
      yorn='n'
      xlen=6.0
      ylen=4.5
      xmin=1.e+37
      xmax=-1.e+37
      ymin=1.e+37
      ymax=-1.e+37
      do 80 i=1,nn
      if(x(i).ge.xmax) xmax=x(i)
      if(x(i).le.xmin) xmin=x(i)
      if(y(i).ge.ymax) ymax=y(i)
      if(y(i).le.ymin) ymin=y(i)
   80 continue
      write(6,*) 'ymin=',ymin,' ymax=',ymax
      if(ylog.eq.'y') write(6,*) '  -y value in log sacle'
      write(6,*) 'xmin=',xmin,' xmax=',xmax
      if(xlog.eq.'y') write(6,*) '  -x value in log sacle'
      write(6,*) ' '
      write(6,*) 'enter ymin,ymax,yinc,xmin,xmax,xinc'
      if(ylog.eq.'y') write(6,*) '       -x should be integer.'
      if(ylog.eq.'y') write(6,*) '       -y should be integer.'
```

```
      read(5,*) y0,y1,yinc,x0,x1,xinc
      xinut=(x1-x0)/xlen
      yinut=(y1-y0)/ylen
 100  continue
      if(kk.ne.1) go to 500
      write(6,*) 'plot axis? (y/n)'
      read(5,5) nory
 5    format(a)
      write(6,*) ' '
      write(6,*) 'wait.  It is processing.'
      if(nory.ne.'y') go to 500
      call plot(xlen,0.0,2)
      call plot(xlen,ylen,2)
      call plot(0.0,ylen,2)
      call plot(0.0,0.0,2)
      call plot(0.0,-0.05,2)
      if(xlog.eq.'y') go to 110
      if(x0.lt.0.0) xshif=-0.25
      if(x0.ge.0.0) xshif=-0.13
      if(x0.ge.10.0) xshif=-0.23
      call number(xshif,-0.17,0.1,x0,0.0,1)
      go to 120
 110  call number(-0.12,-0.18,0.1,10.0,0.0,-1)
      call number(999.,-0.1,0.06,x0,0.0,-1)
 120  continue
      grid=xinc/xinut
      xi=1.
 200  xx=xi*grid
      if(abs(xx).gt.xlen+0.02) go to 250
      call plot(xx,0.0,3)
      call plot(xx,-0.05,2)
      xsym=x0+xinc*xi
      if(xlog.eq.'y') go to 210
      if(xsym.lt.0.0) xshif=-0.25
      if(xsym.ge.0.0) xshif=-0.13
      if(xsym.ge.10.0) xshif=-0.23
      if(xsym.ge.100.0) xshif=-0.33
      call number(xx+xshif,-0.17,0.1,xsym,0.0,1)
      go to 220
 210  continue
      call number(xx-0.12,-0.18,0.1,10.0,0.0,-1)
      call number(999.,-0.1,0.06,xsym,0.0,-1)
 220  continue
      xi = xi+1.
      go to 200
 250    continue
      call symbol(2.5,-0.5,0.16,bet,0.0,11)
      call plot(0.0,0.0,3)
      call plot(-0.05,0.0,2)
      if(ylog.eq.'y') go to 260
      if(y0.lt.0.0) xshif=-0.45
      if(y0.ge.0.0) xshif=-0.35
      if(y0.ge.10.0) xshif=-0.45
```

```
      call number(xshif,-0.05,0.1,y0,0.0,1)
      go to 270
  260 continue
      call number(-0.34,-0.06,0.1,10.0,0.0,-1)
      call number(-0.16,0.02,0.06,y0,0.0,-1)
  270 continue
      grid=yinc/yinut
      xi=1.
  300 xx=xi*grid
      if(abs(xx).gt.ylen+0.02) go to 400
      call plot(0.0,xx,3)
      call plot(-0.05,xx,2)
      xsym=y0+yinc*xi
      if(ylog.eq.'y') go to 310
      if(xsym.lt.0.0) xshif=-0.45
      if(xsym.ge.0.0) xshif=-0.35
      if(xsym.ge.10.0) xshif=-0.45
      if(xsym.ge.100.0) xshif=-0.55
      call number(xshif,xx-0.05,0.1,xsym,0.0,1)
      go to 320
  310 continue
      call number(-0.34,xx-0.06,0.1,10.0,0.0,-1)
      call number(-0.16,xx+0.02,0.06,xsym,0.0,-1)
  320 continue
      xi = xi+1.
      go to 300
  400 continue
      call symbol(-0.55,1.5,0.16,alp,90.0,11)
  500 continue
      do 550 i=1,nn
      if(y(i).gt.y1) y(i)=y1
      if(x(i).gt.x1) x(i)=x1
      if(y(i).lt.y0) y(i)=y0
      if(x(i).lt.x0) x(i)=x0
  550 continue
      x(nn+1)=x0
      x(nn+2)=xinut
      y(nn+1)=y0
      y(nn+2)=yinut
      call line(x,y,nn,1,0,0)
      return
      end
***********
```

```
c       deriv81.f
c
c       f77 -i -I2 deriv81.f -o deriv81
c
c       This program reads the eigenfunctions and derivatives
c       at different depths from an output file of r(1)eigen81.
c
        common/model/ d(500),a(500),b(500),rho(500)
        common/eigfun/ depth(500),ur(500),uz(500),tz(500),tr(500),
     *                 dcda(500),dcdb(500),dcdr(500)
        character*50 names
   5    format(a)
  10    format(///22x,'M',3x,'  D  ',3x,'  A  ',3x,'  B  ',5x,'RHO')
  20    format(20x,i3,1x,4f7.2)
  30    format(20x,i3,8x,3f7.2)
  40    format(////)
  45    format(/////30x,'T(SEC)  =  ',f7.3/)
  50    format(10x,' E   =  ',e11.4,'    C(KM/S) = ',f7.4,
     *         '    U(energy) = ',f7.4)
  51    format(7x,'C(KM/S) = ',f7.4,'  U(energy) = ',f7.4,
     *         '  ALE = ',e11.4)
  55    format(10x,' IO = ',e11.4,'    I1   = ',e11.4,'    I2 = ',
     *         e11.4)
  56    format(7x,'IO = ',e11.4,'    I1 = ',e11.4,'    I2 = ',
     *         e11.4,'    L = ',e11.4)
  60    format(10x,' I3 = ',e11.4,'    ARE = ',e11.4,'    L   = ',
     *         e11.4)
  70    format(/'   M',5x,'DEPTH',5x,'UR',10x,'UZ',10x,'TZ',10x,
     *         'TR',9x,'DCDA',8x,'DCDB',8x,'DCDR'/)
  71    format(/'   M',8x,'DEPTH',6x,'DISP',8x,'STRESS',8x,
     *         'DC/DB',8x,'DC/DR'/)
  75    format(1x,i3,f10.2,7(1x,e11.4))
  76    format(1x,i3,f13.2,4(2x,e11.4))
  80    format(/9x,'STRESSO = ',e11.4)
        write(6,*) 'Rayleigh(1) or Love(2):'
        read(5,*) llrr
        write(6,*) 'enter the input file name: (from r<1>eigen81)'
        read(5,5) names
        open(1,file=names,status='old',form='unformatted')
        rewind 1
        read(1) mmax,(d(i),a(i),b(i),rho(i),i=1,mmax)
        write(6,*) ' '
        write(6,10)
        do 100 i=1,mmax-1
        write(6,20) i,d(i),a(i),b(i),rho(i)
 100    continue
        i=mmax
        write(6,30) i,a(i),b(i),rho(i)
 200    continue
        read(1) ifun,mode,per
        if(ifun.lt.0) go to 500
        if(mode.le.0) go to 200
        write(6,40)
```

```
      omega=6.2831853/per
      do 400 k=1,mode
      read(1) e,ee,c,ugr,xi0,xi1,xi2,xi3,are,flagr
      wvno=omega/c
      write(6,45) per
      go to (210,220),llrr
  210 continue
      write(6,50) e,c,ugr
      write(6,55) xi0,xi1,xi2
      write(6,60) xi3,are,flagr
      go to 250
  220 continue
      write(6,51) c,ugr,are
      write(6,56) xi0,xi1,xi2,flagr
  250 continue
      if(llrr.eq.1) write(6,70)
      if(llrr.eq.2) write(6,71)
      do 300 i=1,mmax
      read(1) depth(i),ur(i),uz(i),tz(i),tr(i),
     *             dcda(i),dcdb(i),dcdr(i)
      if(llrr.eq.1)
     *  write(6,75) i,depth(i),ur(i),uz(i),tz(i),tr(i),
     *              dcda(i),dcdb(i),dcdr(i)
      if(llrr.eq.2)
     *  write(6,76) i,depth(i),ur(i),uz(i),dcdb(i),dcdr(i)
  300 continue
      write(6,80) ee
  400 continue
      go to 200
  500 continue
      stop
      end
***********
```

```
c       wig81.f
c
c       f77 -i -I2 wig81.f -o wig81
c
c       This program generates ten basic types of synthetic
c       spectrum after combining the eigenfunctions from
c       reigen81 and/or eigen81 with the source spectrum.
c
c       The output can be a seismogram which will input to
c       the program gle81 or a spectrum which will input to
c       the program spec81 for plotting.
c
c       The dimension needed has been reduced to the minimum by
c       using the system call.
c       The program 'bigfft' to do fast Fourier transform should
c       exist in the present working directory.
c
c       The maximum point for time histories is 8192.
c       The maximum mode number at any period is 400.
c
        dimension rr(20),tshif(20),np(20)
        common/srctim/ src(500)
        common/ctrl/  ms,sr,si,xmom,r0,t0,np0,kkr,kkl,kkf,ino
        common/resp/  df,bmax,mode,per1,per3(200),xx(20),yy(20)
        character*50  names
 5      format(a)
        open(7,file='t1.d',status='scratch',form='unformatted')
        open(8,file='t2.d',status='scratch',form='unformatted')
        write(6,*)
      *  'BE SURE  bigfft  exist in the present directory.'
        write(6,*) ' '
        write(6,*)
      *   'enter the name of input file: (from reigen81)'
        write(6,*) '(if not use, answer  none )'
        read(5,5) names
        kkr=0
        if(names.eq.'none') go to 100
        kkr=1
        open(1,file=names,status='old',form='unformatted')
 100    write(6,*)
      *   'enter the name of input file: (from leigen81)'
        write(6,*) '(if not use, answer  none )'
        kkl=0
        read(5,5) names
        if(names.eq.'none') go to 120
        kkl=1
        open(2,file=names,status='old',form='unformatted')
 120    continue
        write(6,*)
      *   'seismograms(1), spectrum(2), or both(3) wanted?'
        read(5,*) ityp
        if(ityp.eq.2) go to 140
        write(6,*)
```

```
     *    'enter the name of output file for seismogram:'
      read(5,5) names
      open(3,file=names,status='new',form='unformatted')
      rewind 3
 140  if(ityp.eq.1) go to 160
      write(6,*)
     *    'enter the name of output file for spectrum:'
      read(5,5) names
      open(4,file=names,status='new',form='unformatted')
      rewind 4
      write(6,*) 'what kind of spectrum wanted?'
      write(6,*)
     *    'all modes(1),  fund-high modes separated(2):'
      read(5,*) kkf
 160  continue
      write(6,*) 'enter source seismic moment(in 1.e+20):'
      read(5,*) xmom
      write(6,*) 'enter source type (step:1 bell:2 readin:3):'
      read(5,*) ms
      write(6,*) 'enter dt:'
      read(5,*) dt
      if(ms.eq.2.or.ms.eq.3) call source(ms,dt)
 170  continue
      write(6,*) 'enter station locations- r,tshift,npt:'
      write(6,*)
     *    '(use r,tshift,-npt if no interpolation wanted.'
      write(6,*) ' use -1,0,0 to stop this sequence.  )'
      i=1
 200  continue
      read(5,*) rr(i),tshif(i),np(i)
      if(rr(i).le.0.0) go to 260
      if(np(i).lt.0) go to 250
      npt=np(i)
      npt0=1
 230  continue
      npt0=2*npt0
      if(npt.eq.npt0) go to 250
      npt1=2*npt0
      if(npt.lt.npt1.and.npt.gt.npt0) go to 240
      go to 230
 240  continue
      np(i)=npt0
      write(6,*)
     *    'npt is not in power of 2.  adjust to ',np(i)
 250  continue
      i=i+1
      go to 200
 260  nsta=i-1
      if(ityp.eq.1.or.ityp.eq.3) write(3) dt,nsta,kkr,kkl
      if(ityp.eq.2.or.ityp.eq.3) write(4) dt,nsta,kkr,kkl,kkf
      k0=0
      zero=0.0
      open(9,file='bigfft.d',status='new',form='unformatted')
```

```
      do 400 k1=1,nsta
      k0=k0+1
      r0=rr(k1)
      t0=tshif(k1)
      np0=np(k1)
      in0=1
      if(np0.le.0) in0=2
      np0=abs(np0)
      df=1./(np0*dt)
      if(ms.eq.1) go to 350
c     generate the source spectrum.
      if(k1.ne.1.and.np0.eq.np(k1-1)) go to 350
      isign=-2
      rewind 9
      write(9) np0,dt,df,isign
      do 300 i=1,np0,2
      i1=i+1
      if(i.le.500) write(9) src(i),src(i1)
      if(i.gt.500) write(9) zero,zero
  300 continue
      close(9)
      call system('bigfft',kretn)
      open(9,file='bigfft.d',status='old',form='unformatted')
  350 continue
      itp=ityp+2
      if(ityp.eq.3) itp=3
      ino=in0
      if(itp.eq.3) ino=1
c     the main call.
c
      call main(k0,k1,itp)
      if(ityp.eq.3) call main(k0,k1,4)
  400 continue
      close(1)
      close(2)
      if(ityp.eq.1.or.ityp.eq.3) close(3)
      if(ityp.eq.2.or.ityp.eq.3) close(4)
      close(7,status='delete')
      close(8,status='delete')
      write(6,*) ' '
      write(6,*) 'wig81 finished'
      write(6,*) ' '
      stop
      end
c
c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine main(k0,k1,itp)
      dimension d(80),a(80),b(80),rho(80),qa1(80),qb1(80)
      common/ctrl/  ms,sr,si,xmom,r0,t0,np0,kkr,kkl,kkf,ino
      common/resp/  df,bmax,mode,per1,per3(200),xx(20),yy(20)
   10 format(/2x,'M',5x,'D',7x,'A',7x,'B',6x,'RHO',5x,'QA',
     *          6x,'QB')
```

```
   20  format(i3,6f8.2)
   30  format('source depth =',f7.2/
     *          'no. of periods generated in surface81 =',i5)
       rewind 7
       rewind 8
       k0=k0+100
       kk=kkr+kkl
       do 200 k2=1,kk
       icode=k2
       if(kk.eq.1.and.kkl.eq.1) icode=2
       rewind icode
       if(k0*k2.eq.101) write(6,10)
       read(icode)
     *          mmax,(d(i),a(i),b(i),rho(i),qa1(i),qb1(i),i=1,mmax)
       bmax=b(mmax)
       do 100 i=1,mmax
  100  if(k0*k2.eq.101)
     *          write(6,20) i,d(i),a(i),b(i),rho(i),qa1(i),qb1(i)
       read(icode) nper,dphsrc
       if(k0*k2.eq.101) write(6,30) dphsrc,nper
       if(k0*k2.eq.101.or.k0*k2.eq.201) write(itp) dphsrc,nper
       if(k2.eq.1) write(itp) r0,t0,np0
       write(6,*) ' '
       if(itp.eq.3.and.icode.eq.1)
     *    write(6,*) '   RAYLEIGH seismogram for station # ',k1,
     *               ' is under calculation.'
       if(itp.eq.3.and.icode.eq.2)
     *    write(6,*) '   LOVE seismogram for station # ',k1,
     *               ' is under calculation.'
       if(itp.eq.4.and.icode.eq.1)
     *    write(6,*) '   RAYLEIGH spectrum for station # ',k1,
     *               ' is under calculation.'
       if(itp.eq.4.and.icode.eq.2)
     *    write(6,*) '   LOVE spectrum for station # ',k1,
     *               ' is under calculation.'
c
c      the main routine.
       if(ino.eq.1) call inter(itp,icode)
       if(ino.eq.2) call noint(itp,icode)
  200  continue
c
       rewind 7
       rewind 8
       go to (300,500),itp-2
  300  continue
       if(kkr.eq.1) read(7) per,(xx(i),i=1,16)
       if(kkl.eq.1) read(8) per,(xx(i),i=17,20)
       write(3) (xx(i),i=1,20)
       if(per.le.0.0) go to 600
       go to 300
  500  continue
       if(kkr.eq.1) read(7) per,(xx(i),i=1,16)
       if(kkl.eq.1) read(8) per,(xx(i),i=17,20)
```

```
      write(4) per,(xx(i),i=1,20)
      if(kkf.eq.1.and.per.lt.0.0) return
      if(kkf.eq.1) go to 500
      if(kkr.eq.1) read(7) per,(xx(i),i=1,16)
      if(kkl.eq.1) read(8) per,(xx(i),i=17,20)
      write(4) per,(xx(i),i=1,20)
      if(per.le.0.0) go to 600
      go to 500
  600 continue
      return
      end
c
c  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine inter(itp,icode)
c     This routine sets the end values for interpolation.
      common/dimen/ ur(200,2),dur(200,2),uz(200,2),duz(200,2),
     *              wvno(200,2),ur0(200,2),are(200,2),gamma(200,2)
      common/ctrl/  ms,sr,si,xmom,r0,t0,np0,kkr,kkl,kkf,ino
      common/resp/  df,bmax,mode0,per1,per3(200),
     *               xr(16),xl(4),yr(16),yl(4)
      if(ms.eq.1) go to 90
      rewind 9
      read(9) npx,dtx,dfx,isign
   90 continue
      np2=np0/2+1
      icode1=icode+6
      read(icode) ifunc,mode2,per2
      if(ifunc.lt.0) write(6,*) 'no data in the eigen file.'
      do 200 j=1,mode2
      read(icode) wvno(j,2),ur0(j,2),are(j,2),ugr,gamma(j,2)
      read(icode) ur(j,2),dur(j,2),uz(j,2),duz(j,2)
  200 continue
      itrig=0
      do 800 i=np2,1,-1
      do 300 j=1,16
      xr(j)=0.0
      yr(j)=0.0
  300 continue
      do 310 j=1,4
      xl(j)=0.0
      yl(j)=0.0
  310 continue
      per=-1.0
      if(i.eq.1) go to 750
      if(ms.ne.1) read(9)  sr,si
      per=1./((i-1)*df)
      if(i.eq.2) go to 320
      dper=1./((i-2)*df)-per
  320 continue
      per0=per-0.005*dper
      if(i.eq.np2) per0=per+0.005*dper
  330 if(per0.lt.per2) go to 400
```

```
      if(itrig.eq.2) go to 750
      per1=per2
      mode1=mode2
      read(icode,end=750) ifunc,mode2,per2
      itrig=1
      if(ifunc.gt.0) go to 340
      itrig=2
      go to 330
  340 continue
      do 350 j=1,mode1
      wvno(j,1) = wvno(j,2)
      ur0(j,1)  = ur0(j,2)
      are(j,1)  = are(j,2)
      gamma(j,1) = gamma(j,2)
      ur(j,1)   = ur(j,2)
      dur(j,1)  = dur(j,2)
      uz(j,1)   = uz(j,2)
      duz(j,1)  = duz(j,2)
  350 continue
      do 360 j=1,mode2
      per3(j)=per2
      read(icode) wvno(j,2),ur0(j,2),are(j,2),ugr,gamma(j,2)
      read(icode) ur(j,2),dur(j,2),uz(j,2),duz(j,2)
  360 continue
      go to 330
  400 if(itrig.eq.0) go to 750
      mod=mode1-mode2
      mode0=mode2
  700 continue
      kk=itp-2
      if(itp.eq.4.and.kkf.eq.1) kk=4
      go to (710,720),icode
  710 call excitr(per,kk)
      if(itp.eq.4.and.kkf.eq.2) call excitr(per,3)
      go to 750
  720 call excitl(per,kk)
      if(itp.eq.4.and.kkf.eq.2) call excitl(per,3)
  750 continue
      go to (760,770),icode
  760 write(7) per,xr
      if(itp.eq.4.and.kkf.eq.2) write(7) per,yr
      go to 800
  770 write(8) per,xl
      if(itp.eq.4.and.kkf.eq.2) write(8) per,yl
  800 continue
      return
      end
c
c  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine noint(itp,icode)
      common/dimen/ ur(200,2),dur(200,2),uz(200,2),duz(200,2),
     *              wvno(200,2),ur0(200,2),are(200,2),gamma(200,2)
```

```
      common/ctrl/  ms,sr,si,xmom,r0,t0,np0,kkr,kkl,kkf,ino
      common/resp/  df,bmax,mode,per1,per3(200),
     *                xr(16),xl(4),yr(16),yl(4)
      icode1=icode+6
  300 continue
      do 310 j=1,16
      xr(j)=0.0
      yr(j)=0.0
  310 continue
      do 320 j=1,4
      xl(j)=0.0
      yl(j)=0.0
  320 continue
      read(icode,end=900) ifunc,mode,per
      if(ifunc.le.0) per=-1.0
      if(ifunc.le.0) go to 750
      do 330 j=1,mode
      read(icode) wvno(j,1),ur0(j,1),are(j,1),ugr,gamma(j,1)
      read(icode) ur(j,1),dur(j,1),uz(j,1),duz(j,1)
      wvno(j,2)=wvno(j,1)
      ur0(j,2)=ur0(j,1)
      are(j,2)=are(j,1)
      gamma(j,2)=gamma(j,1)
      ur(j,2)=ur(j,1)
      dur(j,2)=dur(j,1)
      uz(j,2)=uz(j,1)
      duz(j,2)=duz(j,1)
  330 continue
      if(ms.eq.1) go to 650
      np2=np0/2+1
      per1=1./((np2-1)*df)
      if(per.lt.per1) go to 600
      rewind 9
      read(9) npx,dtx,dfx,isign
      read(9) sr1,si1
      j=np2-2
  400 continue
      if(j.eq.0) go to 600
      per2=1./(j*df)
      read(9) sr2,si2
      if(per.gt.per2.and.per.le.per1) go to 500
      per1=per2
      sr2=sr1
      si2=si1
      j=j-1
      go to 400
  500 sr=sr1+(per-per1)/(per2-per1)*(sr2-sr1)
      si=si1+(per-per1)/(per2-per1)*(si2-si1)
      go to 650
  600 write(6,*) 'The period range of source pulse spectrum'
      write(6,*) 'does not cover the period =',per
      write(6,*) 'Now r=',r0,' npt=',np0,' dt=',dtx
      write(6,*) 'and the period range is ',1./((np2-1)*df),
```

```
     *                  '-',1./df
      write(6,*) 'TRY DIFFERENT npt,dt.'
      stop
  650 continue
      kk=itp-2
      if(itp.eq.4.and.kkf.eq.1) kk=4
      go to (710,720),icode
  710 call excitr(per,kk)
      if(itp.eq.4.and.kkf.eq.2) call excitr(per,3)
      go to 750
  720 call excitl(per,kk)
      if(itp.eq.4.and.kkf.eq.2) call excitl(per,3)
  750 continue
      go to (760,770),icode
  760 write(7) per,xr
      if(itp.eq.4.and.kkf.eq.2) write(7) per,yr
      go to 800
  770 write(8) per,xl
      if(itp.eq.4.and.kkf.eq.2) write(8) per,yl
  800 continue
      go to 300
  900 continue
      return
      end
c
c  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine excitr(per,ident)
c     This routine generates the Z and R components of
c     seismogram for
c      1: 45-deg dip-slip source  2: strike-slip source
c      3: dip-slip source         4: explosion source.
c
c     The number 200 in dimension declaration limits the
c       number of mode at a particular frequency not over 200.
c
      dimension dk(4),dkk(4),vz(2,4),vr(2,4)
      common/dimen/ ur1(200,2),dur1(200,2),uz1(200,2),
     *               duz1(200,2),wvno1(200,2),ur01(200,2),
     *               are1(200,2),gamma(200,2)
      common/ctrl/  ms,sr0,si0,xmom,rx,tx,npx,kkr,kkl,kkf,ino
      common/resp/  df,bmax,mode,per1,per3(200),xz(2,4),
     *               xr(2,4),xt(2,2),yz(2,4),yr(2,4),yt(2,2)
      do 90 i=1,2
      do 90 j=1,4
      vz(i,j)=0.0
      vr(i,j)=0.0
   90 continue
      rxx=rx
c     spectra normalized to a distance of 1000 km.
      if(ident.ge.2) rxx=1000.0
      j1=1
      if(ident.eq.3) j1=2
```

```
j2=mode
if(ident.eq.2) j2=1
if(j1.gt.j2) return
do 200 j=j1,j2
rat=0.0
if(ino.eq.1) rat=(per-per1)/(per3(j)-per1)
wvno=wvno1(j,1) + (wvno1(j,2)-wvno1(j,1))*rat
atn=gamma(j,1)*rx
fact1=0.0
if(atn.lt.80.0) fact1=1./exp(atn)
atn=gamma(j,2)*rx
fact2=0.0
if(atn.lt.80.0) fact2=1./exp(atn)
omega=6.2831853/per
v1 = are1(j,1)/sqrt(wvno1(j,1)*rxx)
v2 = are1(j,2)/sqrt(wvno1(j,2)*rxx)
w1 = duz1(j,1)+0.5*wvno1(j,1)*ur1(j,1)
w1 = w1*v1*fact1
u1 = w1*ur01(j,1)
w2 = duz1(j,2)+0.5*wvno1(j,2)*ur1(j,2)
w2 = w2*v2*fact2
u2 = w2*ur01(j,2)
dk(1) = w1+(w2-w1)*rat
dkk(1)= u1+(u2-u1)*rat
w1 = wvno1(j,1)*ur1(j,1)
w1 = w1*v1*fact1
u1 = w1*ur01(j,1)
w2 = wvno1(j,2)*ur1(j,2)
w2 = w2*v2*fact2
u2 = w2*ur01(j,2)
dk(2) = w1+(w2-w1)*rat
dkk(2)= u1+(u2-u1)*rat
w1 = wvno1(j,1)*uz1(j,1)+dur1(j,1)
w1 = w1*v1*fact1
u1 = w1*ur01(j,1)
w2 = wvno1(j,2)*uz1(j,2)+dur1(j,2)
w2 = w2*v2*fact2
u2= w2*ur01(j,2)
dk(3) = w1+(w2-w1)*rat
dkk(3)= u1+(u2-u1)*rat
w1 = duz1(j,1)-wvno1(j,1)*ur1(j,1)
w1 = w1*v1*fact1
u1 = w1*ur01(j,1)
w2 = duz1(j,2)-wvno1(j,2)*ur1(j,2)
w2 = w2*v2*fact2
u2 = w2*ur01(j,2)
dk(4) = w1+(w2-w1)*rat
dkk(4)= u1+(u2-u1)*rat
t0=xmom/2.5066283
t1=omega*tx-wvno*rx-0.7853981632
t2=omega*tx-wvno*rx-2.356194489
ct1=cos(t1)
st1=sin(t1)
```

```
      ct2=cos(t2)
      st2=sin(t2)
      do 100 k=1,4
      ct=ct1
      if(k.eq.3) ct=-st1
      st=st1
      if(k.eq.3) st=ct1
      vz(1,k)=vz(1,k) - dk(k)*t0*ct
      vz(2,k)=vz(2,k) - dk(k)*t0*st
      ct=ct2
      if(k.eq.3) ct=-st2
      st=st2
      if(k.eq.3) st=ct2
      vr(1,k)=vr(1,k) + dkk(k)*t0*ct
      vr(2,k)=vr(2,k) + dkk(k)*t0*st
  100 continue
  200 continue
      if(ms.eq.1) go to 220
      sr=sr0
      si=si0
      go to 240
  220 sr=0.0
      si=-per/6.2831853
  240 continue
      do 300 k=1,4
      t0=vz(1,k)
      vz(1,k)=sr*vz(1,k)-si*vz(2,k)
      vz(2,k)=sr*vz(2,k)+si*t0
      t0=vr(1,k)
      vr(1,k)=sr*vr(1,k)-si*vr(2,k)
      vr(2,k)=sr*vr(2,k)+si*t0
  300 continue
      do 400 i=1,2
      do 400 j=1,4
      if(ident.ne.3) xz(i,j)=vz(i,j)
      if(ident.ne.3) xr(i,j)=vr(i,j)
      if(ident.eq.3) yz(i,j)=vz(i,j)
      if(ident.eq.3) yr(i,j)=vr(i,j)
  400 continue
      return
      end
c
c  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine excitl(per,ident)
c     This routine generates the T component of seismograms
c     for  1: dip-slip source      2: strike-slip source.
c
c     vt(i,j)    i=1,2  real & imag part     j=1,2  source
c
      dimension dk(2),vt(2,2)
      common/dimen/ ut1(200,2),dut1(200,2),dz1(200,2),
     *              duz1(200,2),wvno1(200,2),ur01(200,2),
```

```
     *                  ale1(200,2),gamma(200,2)
      common/ctrl/  ms,sr0,si0,xmom,rx,tx,npx,kkr,kkl,kkf,ino
      common/resp/  df,bmax,mode,per1,per3(200),xx(16),
     *                  xt(2,2),yy(16),yt(2,2)
      do 90 i=1,2
      do 90 j=1,2
      vt(i,j)=0.0
 90   continue
      rxx=rx
      if(ident.ge.2) rxx=1000.0
      j1=1
      if(ident.eq.3) j1=2
      j2=mode
      if(ident.eq.2) j2=1
      if(j1.gt.j2) return
      do 100 j=j1,j2
      rat=0.0
      if(ino.eq.1) rat=(per-per1)/(per3(j)-per1)
      wvno=wvno1(j,1) + (wvno1(j,2)-wvno1(j,1))*rat
      atn=gamma(j,1)*rx
      fact1=0.0
      if(atn.lt.80.0) fact1=1./exp(atn)
      atn=gamma(j,2)*rx
      fact2=0.0
      if(atn.lt.80.0) fact2=1./exp(atn)
      omega=6.2831853/per
      v1 = ale1(j,1)/sqrt(wvno1(j,1)*rxx)
      v2 = ale1(j,2)/sqrt(wvno1(j,2)*rxx)
      w1 = dut1(j,1)*v1*fact1
      w2 = dut1(j,2)*v2*fact2
      dk(1) = w1+(w2-w1)*rat
      w1 = wvno1(j,1)*ut1(j,1)
      w1 = w1*v1*fact1
      w2 = wvno1(j,2)*ut1(j,2)
      w2 = w2*v2*fact2
      dk(2) = w1+(w2-w1)*rat
      t0=xmom/2.5066283
      t1=omega*tx-wvno*rx+0.7853981632
      ct1=cos(t1)
      st1=sin(t1)
      vt(1,1)=vt(1,1) + dk(1)*t0*st1
      vt(2,1)=vt(2,1) - dk(1)*t0*ct1
      vt(1,2)=vt(1,2) + dk(2)*t0*ct1
      vt(2,2)=vt(2,2) + dk(2)*t0*st1
 100  continue
      if(ms.eq.1) go to 120
      sr=sr0
      si=si0
      go to 140
 120  sr=0.0
      si=-per/6.2831853
 140  continue
      do 200 k=1,2
```

```
      t0=vt(1,k)
      vt(1,k)=sr*vt(1,k)-si*vt(2,k)
      vt(2,k)=sr*vt(2,k)+si*t0
  200 continue
      do 300 i=1,2
      do 300 j=1,2
      if(ident.ne.3) xt(i,j)=vt(i,j)
      if(ident.eq.3) yt(i,j)=vt(i,j)
  300 continue
      return
      end
c
c  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine source(ms,dt)
c     enter source time function.
c
      common/srctim/ src(500)
      character*50 names
      do 50 i=1,500
      src(i)=0.0
  50  continue
      go to (100,200),ms-1
  100 write(6,*) 'enter t1 for bell type of source'
      read(5,*) t1
      call pulse(src,500,dt,t1)
      return
  200 write(6,*)
     *  'input source time function from here(1) or file(2)?'
      read(5,*) i0
      if(i0.eq.1) go to 250
      write(6,*) 'enter file name storing source function:'
      read(5,5) names
  5   format(a)
      open(9,file=names,status='old',form='formatted')
      rewind 9
      j=1
  220 continue
      read(9,*,end=230) src(j)
      j=j+1
      if(j.eq.500)
     *   write(6,*) 'source timeseries too long!  <500'
      if(j.eq.500) go to 230
      go to 220
  230 continue
      close(9)
      return
  250 continue
      write(6,*)
     *   'enter source time function for every (i-1)*dt:'
      write(6,*) '(use -1000.0 to stop)'
      j=1
  260 continue
```

```
      read(5,*) src(j)
      j=j+1
      if(src(j).lt.-999.0) go to 280
      go to 260
  280 src(j)=0.0
      return
      end
c
c  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine pulse(f,n,dt,tl)
c     give a source time function
c
      dimension f(1)
      t1 = 0.0
      t2 = t1 + tl
      t3 = t2 + tl
      t4 = t3 + tl
      t5 = t4 + tl
      do 100 i = 1,n
      y = (i-1)*dt
      z = y - t1
      f(i) = 0.0
      if(y.gt.t1) go to 101
      go to 100
  101 if(y.gt.t2) go to 102
      f(i) = (z/tl)*(z/tl)*0.5
      go to 100
  102 if(y.gt.t3) go to 103
      f(i) = - (z/tl)*(z/tl)*0.5 + 2.0*(z/tl) - 1.0
      go to 100
  103 if(y.gt.t4) go to 104
      f(i) = - (z/tl)*(z/tl)*0.5 + 2.0*(z/tl) - 1.
      go to 100
  104 if(y.gt.t5) go to 105
      f(i) = (z/tl)*(z/tl) * 0.5 -4.0 * (z/tl) + 8.0
      go to 100
  105 f(i) = 0.0
  100 continue
c     area of pulse normalized to unity
      do 200 i = 1,n
  200 f(i) = f(i) /(2.*tl)
      return
      end
*********
```

```
c       gle81.f
c
c       f77 -i -I2 gle81.f -o gle81 -lcalcompI2
c
c       This program follows the program wig81 and generates
c       the seismogram after passing through an instrument.
c
c       The instrument response is imposed in the frequency
c       domain.
c
c       The source mechanism should be given here.
c
c       The dimension needed has been reduced to the minimum,
c       and time series point can be as large as 8192.
c       The restriction for this point dimension comes from
c       the FFT program 'bigfft', which should exist in
c       the present working directory.
c
c       -Oct 10, 1981.
c
        dimension jj(5),az(100),baz(100)
        complex data(10),rr,tt,ff
        common/dimen/ x(1026),y(1026)
        common/pltses/ t0,yend,y1,iplt,jctrl,bb,cc
        common/keep/ s1,s2,s3,t1,t2,cos1,sin1,df
        character*2 form(5),bb,cc
        character*50 names
        data form/'Z ','R ','SH','NS','EW'/
    5   format(a)
   10   format(/1x,a2,' component for station at R=',f8.1,
       *                 ' Az=',f6.1,' is being processed.')
   20   format(/1x,'ins=',i2,9x,'peak= ',f7.1,'    iresp= ',
       *   i2/' dt= ',f9.5,'  ndist= ',i2,7x,'dphsrc= ',f6.1/
       *         ' Z,R,SH,N,E:    ',5i2,5x,'  Quake,Expl:   ',i2/
       *       ' dip= ',f5.1,5x,'slip= ',f5.1,5x,'strk= ',f5.1)
   30   format(a2,1x,'component')
        write(6,*) ' '
        write(6,*) 'BE SURE ',
       *  'the program bigfft exist in the present directory,'
        write(6,*)
       *  'and not be used by other similar jobs.'
        write(6,*) ' '
        write(6,*) 'calculate only(1), ',
       *  ' plot only(2), or calculate and plot(3):'
        read(5,*) iplt
        write(6,*) 'enter the input file name:'
        read(5,5) names
        open(1,file=names,status='old',form='unformatted')
        rewind 1
        write(6,*) ' '
        write(6,*) 'store the seismogram data: (y/n)'
        read(5,5) bb
        if(bb.eq.'n') go to 60
```

```
      write(6,*)
     *  'data (x,y) will be stored in the format of (2e15.8).'
      write(6,*)
     *  'enter the file name for storing seismogram data:'
      read(5,5) names
      open(9,file=names,status='new',form='formatted')
      rewind 9
   60 continue
      if(iplt.eq.2) go to 70
      open(2,file='bigfft.d',status='new',
     *                    form='unformatted')
      open(4,file='tmpp.d',status='scratch',
     *                      form='unformatted')
      if(iplt.ne.1) go to 70
      write(6,*) ' '
      write(6,*) 'enter output file name ',
     *     'for storing the ploting data:'
      read(5,5) names
      open(3,file=names,status='new',form='unformatted')
      rewind 3
   70 continue
c
c-120 input control parameters.
c
      jctrl=-1
      if(iplt.eq.1) go to 80
      call plots(0,0,7)
      write(6,*) 'original pen move: (1.8,10.3)'
      read(5,*) x1,y1
      call plot(x1,y1,-3)
      yend=y1
      ipen=1
      write(6,*) 'choose the pen:'
      read(5,*) ipen
      call newpen(ipen)
      if(iplt.eq.2) go to 140
   80 continue
      write(6,*) 'choose the type of instrument'
      write(6,*) '(=0 15-100  WWSSN     =1 30-100 WWSSN'
      write(6,*) ' =2 6824-13 LP SYSTEM =3 6824-2 LP SYSTEM'
      write(6,*) ' =4 WWSSN SP          =5 USGS SP'
      write(6,*) ' =6 CSSN  SP'
      write(6,*) ' =-1 do NOT pass through any instrument )'
      read(5,*) ins
      write(6,*)
     *    'enter the peak value for instrument response:'
      read(5,*) peak
      write(6,*)
     *        'Take integration(-1), derivative(1), or not(0):'
      read(5,*) iresp
      read(1) dt,ndist,kkr,kkl
      kk=kkr+kkl
      read(1) dphsrc,nper
```

```
      write(6,*) ' '
      if(kk.eq.2) write(6,*)
     *    'Both Rayleigh and Love eigens been generated.'
      if(kkr.eq.1.and.kkl.eq.0) write(6,*)
     *    'Only Rayleigh eigens been generated.'
      if(kkr.eq.0.and.kkl.eq.1) write(6,*)
     *    'Only Love eigens been generated.'
      write(6,*) ' '
      write(6,*)
     *     'Which components to be plotted? (Z,R,SH,NS,EW)'
      write(6,*)
     *  'if yes answer 1, no answer 0  (e.g. 1,0,0,1,1)'
      read(5,*) (jj(i),i=1,5)
      if(kkr.ne.0) go to 100
      j=jj(1)+jj(2)
      if(j.eq.0) go to 100
      write(6,*) 'no P-SV wave ready. run again.'
      go to 1000
  100 continue
      if(kkl.ne.0) go to 120
      if(jj(3).eq.0) go to 110
      write(6,*) 'no SH wave ready. run again.'
      go to 1000
  110 continue
  120 continue
c
c-150 input source mechanism.
c
      write(6,*) ' '
      write(6,*)
     *    'no. of epicentral distance generated = ',ndist
      write(6,*) 'source depth = ',dphsrc,'   dt = ',dt
      write(6,*) ' '
      write(6,*) 'earthquake source(1) or explosion(2)?'
      read(5,*) m3
      degrad=3.141592653/180.0
      if(m3.eq.2) go to 130
      write(6,*) 'enter the source mechanism  dip,slip,strk:'
      read(5,*) dip0,slip0,strk0
  130 continue
c
      if(iplt.eq.1) write(3) ins,peak,iresp,dt,ndist,dphsrc,
     *                           jj,m3,dip0,slip0,strk0
      go to 150
  140 continue
      read(1) ins,peak,iresp,dt,ndist,dphsrc,jj,m3,dip0,
     *          slip0,strk0
      write(6,20) ins,peak,iresp,dt,ndist,dphsrc,jj,m3,
     *                dip0,slip0,strk0
  150 continue
c
c-900 the main loop.
c
```

```
      iazm=0
      iaz=-1
c
c     stations at different epicentral diatances.
      do 900 kdist=1,ndist
      iaz=iaz+1
      if(iplt.eq.2) go to 210
      read(1) r0,t0,np0
      write(6,*) ' '
      write(6,*) 'r=',r0,' t0=',t0,' np0=',np0
      write(6,*) ' '
      np2=np0/2+1
      df=1.0/(np0*dt)
      if(iaz.lt.iazm) go to 180
      iaz=-1
      write(6,*) 'enter Az,bAz (in degree):'
      write(6,*) '(use bAz=0.0 to set bAz=Az+180.0'
      write(6,*) ' use -1,-1 to stop)'
      k=1
  160 continue
      read(5,*) az(k),baz(k)
      if(az(k).eq.-1.0.and.baz(k).eq.-1.0) go to 170
      if(baz(k).eq.0.0) baz(k)=az(k)+180.0
      if(baz(k).ge.360.0) baz(k)=baz(k)-360.0
      k=k+1
      go to 160
  170 naz=k-1
      write(6,*)
     * 'how many sets of different distances FOLLOWED will use'
      write(6,*)
     *  'the above azimuth data: (if not know, enter 100)'
      read(5,*) iazm
  180 continue
      write(6,*) ' '
      write(6,*) 'wait.'
      rewind 4
      do 200 k=np2,1,-1
      read(1) (data(i),i=1,10)
      write(4) (data(i),i=1,10)
  200 continue
      if(iplt.eq.1)
     *  write(3) r0,t0,np0,naz,(az(k),baz(k),k=1,naz)
  210 continue
      if(iplt.eq.2)
     *  read(1) r0,t0,np0,naz,(az(k),baz(k),k=1,naz)
      n1=np0/1024
      nm=np0-n1*1024
      n1=n1+1
      if(nm.eq.0) n1=n1-1
c
c     stations at different azimuths.
      do 900 kaz=1,naz
      phi0=az(kaz)
```

```
      phi1=baz(kaz)
      if(iplt.eq.2) go to 310
      phi=phi1*degrad
      cos1=cos(phi)
      sin1=sin(phi)
      if(m3.eq.2) go to 300
c
c-300 dislocation source.
c
      strk1=phi0-strk0
      dip=dip0*degrad
      slip=slip0*degrad
      strk1=strk1*degrad
      strk2=2.*strk1
      dip2=2.*dip
      s1=sin(slip)*sin(dip2)
      s2=-(cos(slip)*sin(dip)*sin(strk2)+
     *        0.5*sin(slip)*sin(dip2)*cos(strk2))
      s3=sin(slip)*cos(dip2)*sin(strk1)-
     *   cos(slip)*cos(dip)*cos(strk1)
      t1=sin(slip)*cos(dip2)*cos(strk1)+
     *   cos(slip)*cos(dip)*sin(strk1)
      t2=cos(slip)*sin(dip)*cos(strk2)-
     *     0.5*sin(slip)*sin(dip2)*sin(strk2)
  300 continue
  310 continue
c
c-500 store the spectrum data in bigfft.d
c
c     different components at the same station.
      do 900 j=1,5
      if(jj(j).eq.0) go to 900
      cc=form(j)
      write(6,10) cc,r0,phi0
      if(iplt.eq.2) go to 650
      rewind 2
      rewind 4
      isign=+2
      write(2) np0,dt,df,isign
      do 500 k=np2,2,-1
      read(4) (data(i),i=1,10)
      freq=(k-1)*df
      go to (400,320),m3
  320 go to (330,340,350,360,370),j
  330 ff=data(4)
      go to 460
  340 ff=data(8)
      go to 460
  350 write(6,*) 'explosive source no SH wave.  run again.'
      go to 1000
  360 ff=-cos1*data(8)
      go to 460
  370 ff=-sin1*data(8)
```

```
      go to 460
  400 go to (410,420,430,440,440), j
  410 ff=s1*data(1)+s2*data(2)+s3*data(3)
      go to 460
  420 ff=s1*data(5)+s2*data(6)+s3*data(7)
      go to 460
  430 ff=t1*data(9)+t2*data(10)
      go to 460
  440 rr=s1*data(5)+s2*data(6)+s3*data(7)
      tt=t1*data(9)+t2*data(10)
      ct1=-cos1
      st1=sin1
      if(j.eq.4) go to 450
      ct1=-sin1
      st1=-cos1
  450 ff=ct1*rr+st1*tt
  460 continue
      call resp(ff,freq,ins,peak,iresp)
      write(2) ff
  500 continue
      ff=cmplx(0.0,0.0)
      write(2) ff
c
c.    system call: perform big fft
      close(2)
      call system('bigfft',kturn)
      open(2,file='bigfft.d',status='old',
     *                            form='unformatted')
      rewind 2
      read(2) npx,dtx,dfx,isx
c
c-600 find overall max,min
c
      ymax=-1.0e+38
      ymin= 1 0e+38
      do 600 k=1,np0,2
      read(2) y1,y2
      if(y1.gt.ymax) ymax=y1
      if(y2.gt.ymax) ymax=y2
      if(y1.lt.ymin) ymin=y1
      if(y2.lt.ymin) ymin=y2
  600 continue
      xmin=0.0
      xmax=(np0-1)*dt
      if(iplt.eq.1) write(3) xmin,xmax,ymin,ymax
      rewind 2
      read(2) npx,dtx,dfx,isx
      if(iplt.eq.1) go to 660
  650 continue
      if(iplt.eq.2) read(1) xmin,xmax,ymin,ymax
c     plot axis
  660 continue
      call seiplt(xmin,xmax,ymin,ymax,xmin1,xmax1)
```

```
      if(bb.eq.'n') go to 680
      write(9,30) cc
      write(9,*)
     *  'dt=',dt,' nn=',np0,' from:',xmin1,' to ',xmax1,' sec'
      if(m3.eq.1) write(9,*)
     *  'source depth=',dphsrc,' dip=',dip0,' slip=',
     *    slip0,' strk=',strk0
      if(m3.eq.2) write(9,*)
     *  'source depth=',dphsrc,' explosion source'
      write(9,*)
     *    'r=',r0,' phi=',phi0,' t0=',t0,' instrument=',ins,
     *    ' peak=',peak
  680 continue
c
c--800 divide the whole length into sub-lenth with 1024 as max.
c
      do 800 k=1,n1
      k0=k
      n0=1024
      if(k.eq.n1.and.nm.ne.0) n0=nm
      go to (700,750,700), iplt
  700 do 720 i=1,n0,2
      read(2) y(i),y(i+1)
  720 continue
      if(iplt.eq.1) write(3) (y(i),i=1,n0)
      go to 760
  750 read(1) (y(i),i=1,n0)
  760 continue
      call seical(k0,n1,n0,dt,r0,xmin1,xmax1)
  800 continue
  900 continue
 1000 continue
      if(iplt.ne.1) call plot(8.0,0.0,999)
      close(1)
      close(2)
      close(3)
      close(4,status='delete')
      close(9)
      write(6,*) ' '
      write(6,*) 'gle81 finished'
      write(6,*) ' '
      stop
      end
c
c  - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine resp(ff,freq,ins,peak,iresp)
c     This routine imposes the instrument response on the
c     seismogram by frequency domain multiplication.
      complex ff
      fr=real(ff)
      fi=aimag(ff)
      if(ins.lt.0) go to 250
```

```
c       ins  = 0   WWSSN 15-100              = 1   WWSSN 30-100
c            = 2   LP SYSTEM 6824-13         = 3   LP SYSTEM 6824-2
c            = 4   WWSSN SP                  = 5   USGS SP
c            = 6   CSSN SP
        go to (110,110,120,120,130,130,130),ins+1
  110   call wwssn(freq,peak,ins,pr,pi)
        go to 200
  120   call lpsys(freq,peak,ins,pr,pi)
        go to 200
  130   call sperd(freq,peak,ins,pr,pi)
  200   continue
        tmp=fr
        fr=fr*pr-fi*pi
        fi=tmp*pi+fi*pr
  250   if(iresp.eq.0) go to 400
        omega=2.*3.141592653*freq
        if(iresp.eq.-1) go to 300
        tmp=fr
        fr=-fi*omega
        fi=tmp*omega
        go to 400
  300   continue
        tmp=fr
        fr=fi/omega
        fi=-tmp/omega
  400   ff=cmplx(fr,fi)
        return
        end
c
c  - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
        subroutine wwssn(freq,peak,ins,xr,xi)
c       ins eq 0   15-100 WWSSN
c       ins eq 1   30-100 WWSSN
c       peak magnifications are 350,420,1400,2800,5600
        we=6.2831853*freq
        index=(peak+1)/375
        if(ins.gt.0) go to 200
  100   continue
        go to(1,2,2,3,3,3,3,4,4,4,4,4,4,4,4,5),index
    1   fmag=278.
        sigma=0.003
        go to 6
    2   fmag=556.0
        sigma=0.013
        go to 6
    3   fmag=1110.
        sigma=0.047
        go to 6
    4   fmag=2190.
        sigma=0.204
        go to 6
    5   fmag=3950.
```

```
      sigma=0.805
    6 zeta=0.93
      zeta1=1.
      wn=.418879
      wn1=.062831853
      go to 300
  200 continue
      go to(10,20,20,30,30,30,30,40,40,40,40,40,40,40,40,50),
     *       index
   10 fmag = 251.9
      sigma = 0.003
      go to 60
   20 fmag = 503.1
      sigma = 0.012
      go to 60
   30 fmag = 1001.5
      sigma = 0.044
      go to 60
   40 fmag = 1941.9
      sigma = 0.195
      go to 60
   50 fmag = 2241.8
      sigma = .767
   60 zeta = 1.5
      zeta1 = 1.0
      wn = .2094395
      wn1 = .062831853
  300 continue
      ar= (we*we-wn*wn)*(we*we-wn1*wn1)-4.*zeta*zeta1*wn*wn1*
     *     (1.-sigma)*we*we
      ai=2.*we*(zeta1*wn1*(wn*wn-we*we)+zeta*wn*
     *    (wn1*wn1-we*we))
      factor = fmag*we*we*we / (ai*ai + ar*ar)
      xr =-ai * factor
      xi =-factor * ar
      return
      end
c
c   - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine lpsys(freq,peak,ins,xr,xi)
c     LRSM response for LP system with filter 6824-2  ins=3
c     LRSM response for LP system with filter 6824-13 ins=2
c     phase response obtained from hilbert transform of amplitude
c     response.  gain normalized to 1.0 at 25 seconds.
      dimension fre(28),p(56),phi(56)
c     the first 28 p and phi are for 6824-2, next
c     28 are for 6824-13
      data fre/.001,.002,.003,.004,.005,.006,.007,.008,.009,
     1.01,.02,.03,.04,.05,.06,.07,.08,.09,.1,.2,.3,.4,.5,.6,
     2.7,.8,.9,1./
      data phi/263.9,257.9,251.9,245.6,239.3,233.1,227.1,221.1,
     1214.6,208.3,134.9,73.2,15.3,-33.2,-71.2,-100.1,-122.4,
```

```
     2-140.0,-153.8,-213.1,-232.6,-242.8,-249.2,-253.0,-256.9,
     3-259.3,-261.2,-262.9, 265.0,259.1,253.4,248.1,242.8,236.9,
     4 230.3,223.4,216.4, 209.7,153.4,102.0, 54.6, 14.4,-16.8,
     5 -40.4,-58.2,-73.3, -85.4,-146.4,-153.9,-155.3,-156.4,
     6-157.7,-159.2,-160.0,-162.3,-163.8/
      data p/.00005,.00040,.00135,.00321,.00625,.01077,.01706,.02546,
     1.03631,.05006,.34117,.73904,1.0000,.97633,.79807,.61417,.46105,
     2.34464,.26315,.03583,.01097,.00468,.00240,.00139,.00087,.00058,
     3.00040,.00029,
     4 .000015,.000391,.00131,.00310,.00609,.01068,.01713,.02556,
     5 .03618,.04848,.28030,.67553,1.0,1.09448,1.0044,.86969,.74511,
     6 .63557,.54818,.12153,.04830,.02741,.01827,.01328,.01017,
     7 .008052,.006545,.005404/
      m = 28
      degrad = 0.01745329
      if(freq.gt.0.5) freq = 0.5
      if(freq.lt.0.005) freq = 0.005
      do 200 i = 1,m
      if(freq.ge.fre(i).and.freq.le.fre(i+1)) go to 160
  200 continue
  160 continue
      if(ins.eq.2)j=i+28
      if(ins.eq.3)j=i
      pf= phi(j)+(phi(j+1)-phi(j))/(fre(i+1)-fre(i))*(freq-fre(i))
      ph = p(j) + (p(j+1) - p(j))/(fre(i+1)-fre(i))*(freq-fre(i))
      pf = pf * degrad
      xr = cos(pf) * ph * peak
      xi = sin(pf) * ph * peak
      return
      end
c
c  - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine sperd(freq,peak,ins,xr,xi)
c     ins = 4  WWSSN SP
c     ins = 5  USGS  SP
c     ins = 6  CSSN  SP
c     The values for WWSSN and USGS from Luh (1977, BSSA, p.950)
c
      double complex ss(20),ww,tt
      double precision xnorm,a(20)
      go to (100,200,300), ins-3
  100 continue
      m=3
      n=5
      xnorm=1.007d+1
      a(6)=1.0d+0
      a(5)=5.684d+0
      a(4)=1.510d+1
      a(3)=2.217d+1
      a(2)=1.846d+1
      a(1)=7.220d+0
      go to 400
```

```
  200 continue
      m=4
      n=10
      xnorm=2.921d+15
      a(11)=1.0d+0
      a(10)=4.315d+2
      a(9) =8.449d+4
      a(8) =7.520d+6
      a(7) =4.032d+8
      a(6) =1.161d+10
      a(5) =2.041d+11
      a(4) =1.482d+12
      a(3) =2.348d+12
      a(2) =1.589d+12
      a(1) =2.665d+11
      go to 400
  300 continue
c     Ts=1.0   Tg=0.23   hs=0.8   hg=0.8   sigma=0.0
      m=3
      n=4
      xnorm=.307321224d+2
      a(5)=1.0d+0
      a(4)=.855652d+1
      a(3)=.310340d+2
      a(2)=.372023d+2
      a(1)=.189036d+2
  400 continue
      ww=dcmplx(0.0d+0,dble(freq))
      ss(1)=ww
      do 500 i=2,n
      ss(i)=ss(i-1)*ww
  500 continue
      tt=dcmplx(a(1),0.0)
      do 600 i=1,n
      tt=tt+a(i+1)*ss(i)
  600 continue
      tt=ss(m)/tt
      xnorm=xnorm*peak
      xr=real(tt)*xnorm
      xi=dimag(tt)*xnorm
      return
      end
c
c   - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine seiplt(xmin,xmax,ymin,ymax,xmini,xmaxi)
c     CALCOMP plot.
c     plot axis only.
      common/pltses/ t0,yend,y1,iplt,jctrl,bb,cc
      common/plott/ xnp1,xnp2,ynp1,ynp2,xr,yr
      character*2 bb,cc
c     xinut: how many unit per inch
c     xseg: how many unit for one segment in time axis
```

```
      if(kctrl.le.jctrl) go to 100
      write(6,*) ' '
      write(6,*) 'Total time axis from ',xmin,' to ',xmax
      write(6,*)
     * 'enter time window to be plotted: (start and end time)'
      write(6,*) ' (use -1,-1 to plot the whole time span)'
      read(5,*) xmin1,xmax1
      if(xmin1.lt.0.0.and.xmax1.lt.0.0) then
           xmin1=xmin
           xmax1=xmax
      endif
      xmn1=abs(xmax1-xmin1)
      if(iplt.eq.1) go to 80
      write(6,*)
     * 'how many inches for time axis,  (total= ',xmn1,'sec)'
      write(6,*) 'how many inches for amplitude axis,'
      write(6,*)
     * 'and how many units for one segment? (as 5.0,0.8,10.0)'
      read(5,*) xl,yl,xseg
80    continue
      write(6,*)
     * 'how many plots FOLLOWED will use above numbers?'
      write(6,*) '(if not know, enter 100)'
      read(5,*) jctrl
      kctrl=0
100   continue
      kctrl=kctrl+1
      if(iplt.eq.1) return
      ymxx=ymax
      if(abs(ymin).gt.ymax) ymxx=abs(ymin)
      yend1=yend-yl-0.68
      if(yend1.lt.0.5) call plot(8.5,yl-yend,-3)
      yend=yend1
      if(yend1.lt.0.5) yend=yl-yl-0.68
      call plot(0.0,-yl-0.68,-3)
      xinut=xmn1/xl
      ii=xmin1/xseg
      if(xmin1.lt.0.0) ii=ii-1
      x0=xseg*ii
      xnp1=x0
      xnp2=xinut
      ynp1=-ymxx
      ynp2=2.*ymxx/yl
      power=alog10(ymxx)
      if(power.lt.0.) power=power-1.
      power=aint(power)
      yy=ymxx/10 **power
      call plot(-0.16,0.0,3)
      call plot(-0.1,0.0,2)
      call plot(-0.1,yl,2)
      call plot(-0.16,yl,2)
      call symbol(-0.175,-0.025,0.09,1h-,90.,1)
      call number(999.,999.,0.09,yy,90.,1)
```

```
      call number(999.,yl-0.19,0.09,yy,90.,1)
      call symbol(-0.35,yl*0.25,0.11,3h*10,90.,3)
      call number(-0.4,999.,0.06,power,90.,-1)
      call symbol(-0.8,yl*0.45,0.12,cc,0.0,2)
      xr=xl+0.03
      yr=0.5*yl-0.05
      xlen=abs(xmax1-x0)/xinut
      grid=xseg/xinut
      call plot(0.0,-0.1,3)
      call plot(0.0,-0.14,2)
      if(x0.lt.0.0) xshif=-0.15
      if(x0.ge.0.0) xshif=-0.03
      if(x0.ge.10.0) xshif=-0.08
      if(x0.ge.100.0) xshif=-0.11
      call number(xshif,-0.25,0.08,x0,0.0,-1)
      call plot(0.0,-0.1,3)
      xi=1.
  400 xx=xi*grid
      call plot(xx,-0.1,2)
      call plot(xx,-0.14,2)
      xsym=x0+xseg*xi
      if(xsym.lt.0.0) xshif=-0.15
      if(xsym.ge.0.0) xshif=-0.03
      if(xsym.ge.10.0) xshif=-0.08
      if(xsym.ge.100.0) xshif=-0.11
      if(xsym.ge.1000.0) xshif=-0.15
      call number(xx+xshif,-0.25,0.08,xsym,0.0,-1)
      call plot(xx,-0.1,3)
      xi = xi+1.
      if(abs(xx).gt.xlen) go to 500
      go to 400
  500 continue
      if(t0.le.0.0) go to 520
      call symbol(xx/2.3,-0.45,0.14,'T-',0.0,2)
      call number(999.,999.,0.1,t0,0.0,2)
      go to 600
  520 call symbol(xx/2.05,-0.45,0.14,'T',0.0,1)
  600 continue
      call plot(0.0,0.0,3)
      return
      end
c
c - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine seical(j0,n1,nn,dt,r0,xmin1,xmax1)
c     plot seismogram.
      common/dimen/ x(1026),y(1026)
      common/plott/ xnp1,xnp2,ynp1,ynp2,xr,yr
      common/pltses/ t0,yend,y1,iplt,jctrl,bb,cc
      character*2 bb,cc
      i0=(j0-1)*1024+1
      i00=i0+1023
      if(nn.lt.1024) i00=i0+nn-1
```

```
      x0=(i0-1)*dt
      x00=(i00-1)*dt
      if(x0.le.xmin1.and.x00.le.xmin1) go to 500
      if(x0.ge.xmax1.and.x00.ge.xmax1) go to 500
      do 100 i=1,nn
      x(i)=x0+(i-1)*dt
 100  continue
      do 200 i=1,nn
      if(x(i).ge.xmin1) go to 250
 200  continue
 250  continue
      mm=i-1
      do 300 i=1,nn
      if(x(i).gt.xmax1) go to 350
 300  continue
 350  continue
      mm1=i-1
      kk=mm1-mm
      do 400 i=1,kk
      j=mm+i
      x(i)=x(j)
      y(i)=y(j)
 400  continue
      if(iplt.eq.1) go to 420
      x(kk+1)=xnp1
      x(kk+2)=xnp2
      y(kk+1)=ynp1
      y(kk+2)=ynp2
      call line(x,y,kk,1,0,0)
 420  continue
      if(bb.eq.'n') go to 500
      do 450 i=1,kk
      write(9,10) x(i),y(i)
 10   format(2e15.8)
 450  continue
      if(iplt.eq.1) return
 500  continue
      if(j0.eq.n1) call number(xr,yr,0.1,r0,0.0,-1)
      return
      end
*********
```

```
c       program spec81.f
c
c       f77 -i -I2 spec81.f -o spec81 -lcalcompI2
c
c       This program follows the program wig81 and plots the
c       spectrum after passing through an instrument.
c
c       The source mechanism is included here.
c       The whole processes are treated in the freq domain.
c
c       The dimension needed has been reduced to the minimum,
c       and the number of series point is not limitted.
c
c       -NOV 1, 1981
c
        dimension jj(5),az(100),baz(100)
        complex data(10,2),rr,tt,ff
        common/dimen/ x(1026),y(1026)
        common/ctrl/ ictrl,size,xy(6)
        character*2 form(5),cc,yorn
        character*50 names
        data form/'Z ','R ','SH','NS','EW'/
   5    format(a)
   10   format(/1x,a2,' component for station at R=',f8.1,' Az=',
     *          f6.1,' is being processed.')
   20   format(/1x,'ins=',i2,9x,'peak= ',f7.1,'    iresp= ',i2/
     *          ' dt= ',f9.5,'  ndist= ',i2,7x,'dphsrc= ',f6.1/
     *          ' Z,R,SH,N,E:    ',5i2,5x,'  Quake,Expl:   ',i2/
     *          ' Spectrum Plot:',i2/
     *          ' dip= ',f5.1,5x,'slip= ',f5.1,5x,'strk= ',f5.1)
        write(6,*) ' '
        write(6,*) 'calculation only(1), plot only(2), or both(3):'
        read(5,*) iplt
        write(6,*) ' '
        write(6,*) 'enter the input file name:'
        read(5,5) names
        if(iplt.ne.2)
     *   open(1,file=names,status='old',form='unformatted')
        if(iplt.eq.2)
     *   open(9,file=names,status='old',form='formatted')
        rewind 1
        rewind 9
        if(iplt.eq.2) go to 60
        open(2,file='tmp0.d',status='scratch',form='unformatted')
        open(4,file='tmpp.d',status='scratch',form='unformatted')
        if(iplt.ne.1) go to 60
        write(6,*) 'enter the output file name:'
        read(5,5) names
        open(3,file=names,status='new',form='formatted')
        rewind 3
   60   continue
c
c-120 input control parameters.
```

```
c
      if(iplt.eq.1) go to 70
      call plots(0,0,7)
      write(6,*) 'original pen move: (2.0,2.0)'
      read(5,*) x1,y1
      call plot(x1,y1,-3)
      ictrl=0
      write(6,*) 'choose the pen:'
      read(5,*) ipen
      call newpen(ipen)
      write(6,*) 'enter size (e.g. 1.0):'
      read(5,*) size
      call factor(size)
      write(6,*) 'plot the observed data? (y/n):'
      read(5,5) yorn
      if(yorn.eq.'n') go to 70
      write(6,*) 'enter the file name of observed data:'
      read(5,5) names
      open(8,file=names,status='old',form='formatted')
      rewind 8
   70 continue
      if(iplt.eq.2) go to 140
      write(6,*) 'X-axis to be period(1) or freq(2):'
      read(5,*) iaxis
      if(iplt.eq.2) go to 140
      write(6,*) 'choose the type of instrument'
      write(6,*) '(=0 15-100  WWSSN      =1 30-100 WWSSN'
      write(6,*) ' =2 6824-13 LP SYSTEM =3 6824-2 LP SYSTEM'
      write(6,*) ' =4 WWSSN SP              =5 USGS SP'
      write(6,*) ' =6 CSSN  SP'
      write(6,*) ' =-1 do NOT pass through any instrument )'
      read(5,*) ins
      write(6,*)
     *    'enter the peak value for instrument response:'
      read(5,*) peak
      write(6,*)
     *         'take the integration(-1), derivative(1), or not(0)'
      read(5,*) iresp
      read(1) dt,ndist,kkr,kkl,kkf
      kk=kkr+kkl
      read(1) dphsrc,nper
      write(6,*) ' '
      if(kk.eq.2) write(6,*)
     *     'Both Rayleigh and Love eigens been generated.'
      if(kkr.eq.1.and.kkl.eq.0) write(6,*)
     *     'Only Rayleigh eigens been generated.'
      if(kkr.eq.0.and.kkl.eq.1) write(6,*)
     *     'Only Love eigens been generated.'
      write(6,*) ' '
      write(6,*)
     *    'Which components to be plotted? (Z,R,SH,NS,EW)'
      write(6,*)
     *    'if yes answer 1, no answer 0  (e.g. 1,0,0,1,1)'
```

```
      read(5,*) (jj(i),i=1,5)
      if(kkr.ne.0) go to 100
      j=jj(1)+jj(2)
      if(j.eq.0) go to 90
      write(6,*) 'no P-SV wave ready.  run again.'
      go to 1000
  90  continue
  100 if(kkl.ne.0) go to 120
      if(jj(3).eq.0) go to 110
      write(6,*) 'no SH wave ready.  run again.'
      go to 1000
  110 continue
  120 continue
c
c-130 input source mechanism.
c
      write(6,*) ' '
      write(6,*) 'no. of distance = ',ndist,'  dt=',dt
      write(6,*)
     *   'source depth = ',dphsrc,' spectrum type = ',kkf
      write(6,*) 'earthquake source(1) or explosion(2)?'
      read(5,*) m3
      degrad=3.141592653/180.0
      if(m3.eq.2) go to 130
      write(6,*) 'enter the source mechanism  dip,slip,strk:'
      read(5,*) dip0,slip0,strk0
  130 continue
c
      if(iplt.eq.1) write(3,*) 'control parameter:'
      if(iplt.eq.1) write(3,*) ins,peak,iresp,dt,ndist,dphsrc,
     *                           kkf,jj,m3,dip0,slip0,strk0
      go to 150
  140 continue
      read(9,5) names
      read(9,*) ins,peak,iresp,dt,ndist,dphsrc,kkf,jj,m3,
     *             dip0,slip0,strk0
      write(6,20) ins,peak,iresp,dt,ndist,dphsrc,jj,m3,kkf,
     *               dip0,slip0,strk0
  150 continue
c
c-900 the main loop.
c
      iazm=0
      iaz=-1
      do 900 kdist=1,ndist
      iaz=iaz+1
      if(iplt.eq.2) go to 210
      read(1) r0,t0,np0
      write(6,*) ' '
      write(6,*) 'r=',r0,' t0=',t0,' np0=',np0
      write(6,*) ' '
      if(iaz.lt.iazm) go to 175
      iaz=-1
```

```
      write(6,*) 'enter Az,bAz (in degree):'
      write(6,*) '(use bAz=0.0 to set bAz=Az+180.0'
      write(6,*) ' use -1,-1 to stop)'
      k=1
  160 continue
      read(5,*) az(k),baz(k)
      if(az(k).eq.-1.0.and.baz(k).eq.-1.0) go to 170
      if(baz(k).eq.0.0) baz(k)=az(k)+180.0
      if(baz(k).ge.360.0) baz(k)=baz(k)-360.0
      k=k+1
      go to 160
  170 naz=k-1
      write(6,*)
     *   'how many sets of different distances (r) FOLLOWED'
      write(6,*) ' will use ',
     *   'the above azimuth data: (if not know, enter 100)'
      read(5,*) iazm
  175 continue
      write(6,*) ' '
      write(6,*) 'wait'
      rewind 4
      k=1
  180 continue
      do 190 j=1,kkf
      read(1) per,(data(i,j),i=1,10)
  190 continue
      if(per.le.0.0) go to 200
      write(4) per,((data(i,j),i=1,10),j=1,kkf)
      k=k+1
      go to 180
  200 continue
      np2=k-1
      if(iplt.eq.1)
     *   write(3,*) 'r,t0,npt,nstation,az1,baz1,az2,baz2...:'
      if(iplt.eq.1)
     *   write(3,*) r0,t0,np0,naz,(az(k),baz(k),k=1,naz)
  210 continue
      if(iplt.eq.2) read(9,5) names
      if(iplt.eq.2)
     *   read(9,*) r0,t0,np0,naz,(az(k),baz(k),k=1,naz)
      do 900 kaz=1,naz
      phi0=az(kaz)
      phi1=baz(kaz)
      if(iplt.eq.2) go to 310
      phi=phi1*degrad
      cos1=cos(phi)
      sin1=sin(phi)
      if(m3.eq.2) go to 300
c
c-300 dislocation source.
c
      strk1=phi0-strk0
      dip=dip0*degrad
```

```
      slip=slip0*degrad
      strk1=strk1*degrad
      strk2=2.*strk1
      dip2=2.*dip
      s1=sin(slip)*sin(dip2)
      s2=-(cos(slip)*sin(dip)*sin(strk2)+
     *        0.5*sin(slip)*sin(dip2)*cos(strk2))
      s3=sin(slip)*cos(dip2)*sin(strk1)-
     *   cos(slip)*cos(dip)*cos(strk1)
      t1=sin(slip)*cos(dip2)*cos(strk1)+
     *   cos(slip)*cos(dip)*sin(strk1)
      t2=cos(slip)*sin(dip)*cos(strk2)-
     *      0.5*sin(slip)*sin(dip2)*sin(strk2)
  300 continue
  310 continue
c
      do 900 j=1,5
      if(jj(j).eq.0) go to 900
      cc=form(j)
      write(6,10) cc,r0,phi0
      do 900 kf=1,kkf
      if(iplt.eq.2) go to 640
      rewind 2
      rewind 4
      kzero=0
      do 500 k=1,np2
      read(4) per,((data(i,j0),i=1,10),j0=1,kkf)
      freq=1./per
      go to (400,320),m3
  320 go to (330,340,350,360,370),j
  330 ff=data(4,kf)
      go to 460
  340 ff=data(8,kf)
      go to 460
  350 write(6,*) 'explosive source no SH wave. run again.'
      go to 1000
  360 ff=-cos1*data(8,kf)
      go to 460
  370 ff=-sin1*data(8,kf)
      go to 460
  400 go to (410,420,430,440,440),j
  410 ff=s1*data(1,kf)+s2*data(2,kf)+s3*data(3,kf)
      go to 460
  420 ff=s1*data(5,kf)+s2*data(6,kf)+s3*data(7,kf)
      go to 460
  430 ff=t1*data(9,kf)+t2*data(10,kf)
      go to 460
  440 rr=s1*data(5,kf)+s2*data(6,kf)+s3*data(7,kf)
      tt=t1*data(9,kf)+t2*data(10,kf)
      ct1=-cos1
      st1=sin1
      if(j.eq.4) go to 450
      ct1=-sin1
```

```
      st1=-cos1
  450 ff=ct1*rr+st1*tt
  460 continue
      call resp(ff,freq,ins,peak,iresp)
      gg=sqrt(real(ff)*real(ff)+aimag(ff)*aimag(ff))
      if(gg.le.1.e-35) kzero=kzero+1
      if(gg.le.1.e-35) go to 500
      write(2) freq,gg
  500 continue
      npp=np2-kzero
c-600 find the max,min
      rewind 2
      ymax=-1.0e+38
      ymin= 1.0e+38
      xmax=-1.0e+38
      xmin= 1.0e+38
      do 600 k=1,npp
      read(2) freq,gg
      if(gg.gt.ymax) ymax=gg
      if(gg.lt.ymin) ymin=gg
      if(freq.gt.xmax) xmax=freq
      if(freq.lt.xmin) xmin=freq
  600 continue
      tmp=xmin
      if(iaxis.eq.1) xmin=1./xmax
      if(iaxis.eq.1) xmax=1./tmp
      rewind 2
      if(iplt.ne.1) go to 640
      write(3,*) 'npt0,xmin,xmax,ymin,ymax,T/f:'
      write(3,*) npp,xmin,xmax,ymin,ymax,iaxis
      if(iaxis.eq.1) write(3,*) '  period          amp'
      if(iaxis.eq.2) write(3,*) '  freq            amp'
      go to 660
  640 continue
      if(iplt.ne.2) go to 650
      read(9,5) names
      read(9,*) npp,xmin,xmax,ymin,ymax,iaxis
      read(9,5) names
  650 continue
c     plot axis
      if(kf.eq.1) call sppplt(xmin,xmax,ymin,ymax,cc,iaxis)
      if(kf.eq.1.and.yorn.eq.'y') call obs(iaxis)
  660 continue
      n1=npp/1024
      nm=npp-n1*1024
      n1=n1+1
      if(nm.eq.0) n1=n1-1
c
c-800 divide the whole length into sub-lenth with 1024 as max.
      do 800 k=1,n1
      n0=1024
      if(k.eq.n1.and.nm.ne.0) n0=nm
      if(iplt.eq.2) go to 750
```

```
      do 700 i=1,n0
      read(2) x(i),y(i)
      if(iaxis.eq.1) x(i)=1./x(i)
  700 continue
      if(iplt.eq.1) write(3,40) (x(i),y(i),i=1,n0)
      if(iplt.eq.1) go to 800
  750 continue
      if(iplt.eq.2) read(9,40) (x(i),y(i),i=1,n0)
  40  format(2e13.7)
      call sppcal(n0)
  800 continue
  900 continue
  1000          continue
      if(iplt.ne.1) call plot(8.0,0.0,999)
      close(1)
      close(2,status='delete')
      close(3)
      close(4,status='delete')
      if(yorn.eq.'y') close(8)
      if(iplt.eq.2) close(9)
      write(6,*) ' '
      write(6,*) 'spec81 finished'
      write(6,*) ' '
      stop
      end
c
c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine resp(ff,freq,ins,peak,iresp)
c     This routine imposes the instrument response on the
c     seismogram by frequency domain multiplication.
      complex ff
      fr=real(ff)
      fi=aimag(ff)
      if(ins.lt.0) go to 250
c     ins = 0  WWSSN 15-100            = 1  WWSSN 30-100
c         = 2  LP SYSTEM 6824-13       = 3  LP SYSTEM 6824-2
c         = 4  WWSSN SP                = 5  USGS SP
c         = 6  CSSN SP
      go to (110,110,120,120,130,130,130),ins+1
  110 call wwssn(freq,peak,ins,pr,pi)
      go to 200
  120 call lpsys(freq,peak,ins,pr,pi)
      go to 200
  130 call sperd(freq,peak,ins,pr,pi)
  200 continue
      tmp=fr
      fr=fr*pr-fi*pi
      fi=tmp*pi+fi*pr
  250 if(iresp.eq.0) go to 400
      omega=2.*3.141592653*freq
      if(iresp.eq.-1) go to 300
      tmp=fr
```

```
      fr=-fi*omega
      fi=tmp*omega
      go to 400
  300 continue
      tmp=fr
      fr=fi/omega
      fi=-tmp/omega
  400 ff=cmplx(fr,fi)
      return
      end
c
c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
        subroutine wwssn(freq,peak,ins,xr,xi)
c     ins eq 0  15-100 wwssn
c     ins eq 1 30-100 wwssn
c     peak magnification is fixed at 350,700,1400,2800,5600
        we=6.2831853*freq
        index=(peak+1.)/375.
        if(ins.gt.0) go to 200
        go to (1,2,2,3,3,3,3,4,4,4,4,4,4,4,4,5),index
      1 fmag=278.
        sigma=0.003
        go to 6
      2 fmag=556.0
        sigma=0.013
        go to 6
      3 fmag=1110.0
        sigma=0.047
        go to 6
      4 fmag=2190.0
        sigma=0.204
        go to 6
      5 fmag=3950.0
        sigma=0.805
      6 zeta=0.93
        zeta1=1.0
        wn=.418879
        wn1=.062831853
        go to 300
    200 continue
        go to (10,20,20,30,30,30,30,40,40,40,40,40,40,40,40,50),index
     10 fmag=251.9
        sigma=0.003
        go to 60
     20 fmag=503.1
        sigma=0.012
        go to 60
     30 fmag=1001.5
        sigma=0.044
        go to 60
     40 fmag=1941.9
        sigma=0.195
```

```
      go to 60
   50 fmag=2241.8
      sigma=0.767
   60 zeta=1.5
      zeta1=1.0
      wn=.2094395
      wn1=.062831853
  300 continue
      ar= (we*we-wn*wn)*(we*we-wn1*wn1)-4.*zeta*zeta1*wn*wn1*
     1     (1.-sigma)*we*we
      ai=2.*we*(zeta1*wn1*(wn*wn-we*we)+zeta*wn*(wn1*wn1-we*we))
      factor = fmag*we*we*we / (ai*ai + ar*ar)
      xr =-ai * factor
      xi =-ar * factor
      return
      end
c
c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine lpsys(freq,peak,ins,xr,xi)
c     lrsm response for lp system with filter 6824-2  ins=3
c     lrsm response for lp system with filter 6824-13 ins=2
c     phase response obtained from hilbert transform of amplitude
c     response.  gain normalized to 1.0 at 25 seconds.
      dimension fre(28),p(56),phi(56)
c     the first 28 p and phi are for 6824-2, next
c     28 are for 6824-13
      data fre/.001,.002,.003,.004,.005,.006,.007,.008,.009,
     1.01,.02,.03,.04,.05,.06,.07,.08,.09,.1,.2,.3,.4,.5,.6,
     2.7,.8,.9,1./
      data phi/263.9,257.9,251.9,245.6,239.3,233.1,227.1,221.1,
     1214.6,208.3,134.9,73.2,15.3,-33.2,-71.2,-100.1,-122.4,
     2-140.0,-153.8,-213.1,-232.6,-242.8,-249.2,-253.0,-256.9,
     3-259.3,-261.2,-262.9, 265.0,259.1,253.4,248.1,242.8,236.9,
     4 230.3,223.4,216.4, 209.7,153.4,102.0, 54.6, 14.4,-16.8,
     5 -40.4,-58.2,-73.3, -85.4,-146.4,-153.9,-155.3,-156.4,
     6-157.7,-159.2,-160.0,-162.3,-163.8/
      data p/.00005,.00040,.00135,.00321,.00625,.01077,.01706,.02546,
     1.03631,.05006,.34117,.73904,1.0000,.97633,.79807,.61417,.46105,
     2.34464,.26315,.03583,.01097,.00468,.00240,.00139,.00087,.00058,
     3.00040,.00029,
     4 .000015,.000391,.00131,.00310,.00609,.01068,.01713,.02556,
     5 .03618,.04848,.28030,.67553,1.0,1.09448,1.0044,.86969,.74511,
     6 .63557,.54818,.12153,.04830,.02741,.01827,.01328,.01017,
     7 .008052,.006545,.005404/
       m = 28
       degrad = 0.01745329
       if(freq.gt.0.5) freq = 0.5
       if(freq.lt.0.005) freq = 0.005
       do 200 i = 1,m
       if(freq.ge.fre(i).and.freq.le.fre(i+1)) go to 160
  200 continue
  160 continue
```

```
      if(ins.eq.2)j=i+28
      if(ins.eq.3)j=i
      pf= phi(j)+(phi(j+1)-phi(j))/(fre(i+1)-fre(i))*(freq-fre(i))
      ph = p(j) + (p(j+1) - p(j))/(fre(i+1)-fre(i))*(freq-fre(i))
      pf = pf * degrad
      xr = cos(pf) * ph * peak
      xi = sin(pf) * ph * peak
      return
      end
c
c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine sperd(freq,peak,ins,xr,xi)
c     ins = 4  WWSSN SP
c     ins = 5  USGS  SP
c     ins = 6  CSSN  SP
c     The values for WWSSN and USGS from Luh (1977, BSSA, p.950)
c
      double complex ss(20),ww,tt
      double precision xnorm,a(20)
      go to (100,200,300), ins-3
  100 continue
      m=3
      n=5
      xnorm=1.007d+1
      a(6)=1.0d+0
      a(5)=5.684d+0
      a(4)=1.510d+1
      a(3)=2.217d+1
      a(2)=1.846d+1
      a(1)=7.220d+0
      go to 400
  200 continue
      m=4
      n=10
      xnorm=2.921d+15
      a(11)=1.0d+0
      a(10)=4.315d+2
      a(9) =8.449d+4
      a(8) =7.520d+6
      a(7) =4.022d+8
      a(6) =1.161d+10
      a(5) =2.041d+11
      a(4) =1.482d+12
      a(3) =2.348d+12
      a(2) =1.589d+12
      a(1) =2.665d+11
      go to 400
  300 continue
c     Ts=1.0  Tg=0.23  hs=0.8  hg=0.8  sigma=0.0
      m=3
      n=4
      xnorm=.307321224d+2
```

```
      a(5)=1.0d+0
      a(4)=.855652d+1
      a(3)=.310340d+2
      a(2)=.372023d+2
      a(1)=.189036d+2
  400 continue
      ww=dcmplx(0.0d+0,dble(freq))
      ss(1)=ww
      do 500 i=2,n
      ss(i)=ss(i-1)*ww
  500 continue
      tt=dcmplx(a(1),0.0)
      do 600 i=1,n
      tt=tt+a(i+1)*ss(i)
  600 continue
      tt=ss(m)/tt
      xnorm=xnorm*peak
      xr=real(tt)*xnorm
      xi=dimag(tt)*xnorm
      return
      end
c
c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
         subroutine obs(iaxis)
      common/ctrl/ ictrl,size,xy(6)
         read(8,10) n
    10 format (5x,i5)
         do 100 i=1,n
         inteq=3
         read(8,20) t,y,m
    20 format (f10.8,8x,f12.6,25x,i5)
         if(iaxis.eq.2) t=1./t
         if (y.lt.xy(3)) go to 100
         x=alog10(t)
         x=(x-xy(1))/xy(5)
         y=(y-xy(3))/xy(6)
         if (m.ne.1) inteq=1
      call symbol(x,y,0.07,inteq,0.0,-1)
   100 continue
         return
         end
c
c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine sppplt(xmin,xmax,ymin,ymax,cc,iaxis)
      common/ctrl/  ictrl,size,xy(6)
      character*2 cc,ans
      character*50 names
      ictrl=ictrl+1
      if(ictrl.eq.1) go to 100
      write(6,*) 'enter new origin for the next figure w.r.t.'
      write(6,*) 'the origin of the present figure (8.5,0.0):'
```

```
      read(5,*) x0,y0
      x0=x0/size
      y0=y0/size
      call plot(x0,y0,-3)
100   continue
      write(6,*) 'enter the station identification:'
      read(5,5) names
5     format(a)
      if(xmin.lt.1.e-30) xmin=1.e-30
      if(ymin.lt.1.e-30) ymin=1.e-30
      xy(1)=alog10(xmin)
      xy(2)=alog10(xmax)
      xy(3)=alog10(ymin)
      xy(4)=alog10(ymax)
      do 150 i=1,4
      k=int(xy(i))
      t=k
      if(xy(i).lt.0.0) go to 140
      tt=xy(i)-t
      xy(i)=t
      if(i.eq.1.or.i.eq.3) go to 150
      if(tt.gt.0.001) xy(i)=t+1.0
      go to 150
140   tt=t-xy(i)
      xy(i)=t
      if(i.eq.2.or.i.eq.4) go to 150
      if(tt.gt.0.001) xy(i)=t-1.0
150          continue
      write(6,*)
     *    'xmin= ',xmin,'xmax= ',xmax,'ymin= ',ymin,'ymax= ',ymax
      i1=xy(1)
      i2=xy(2)
      i3=xy(3)
      i4=xy(4)
      write(6,*)
     *    'Log-Log range-  X:',i1,' to ',i2,'  Y:',i3,' to ',i4
      write(6,*) 'give the range for spectrum plotting? (y/n)'
      read(5,5) ans
      if(ans.eq.'n') go to 190
      write(6,*)
     *    'enter your range for X-axis, then for Y-axis: (0,2,-2,3)'
      read(5,*) xy(1),xy(2),xy(3),xy(4)
190   continue
      n1=int(xy(2)-xy(1))
      n2=int(xy(4)-xy(3))
      write(6,*) 'enter the length of x- and y-axis: (5,3.75)'
      read(5,*) xl,yl
      write(6,*) 'plot the frame? (y/n):'
      read(5,5) ans
      if(ans.eq.'n') go to 360
      xl2=0.5*xl
      yl2=0.5*yl
      xseg=xl/n1
```

```
      yseg=yl/n2
      call plot(xl2,0.0,2)
      call plot(xl ,0.0,2)
      call plot(0.0,yl ,3)
      call plot(0.0,yl2,2)
      call plot(0.0,0.0,2)
      do 300 i=1,n1+1
      xt=xseg*(i-1)
      call number(xt-0.09,-0.2,0.1,10.,0.0,-1)
      xp=xy(1)+i-1.
      call number(999.,-0.12,0.06,xp,0.0,-1)
      if(i.eq.n1+1) go to 300
      do 300 j=2,10
      xtt=xt+xseg*alog10(real(j))
      call plot(xtt,0.0,3)
      ytt=0.04
      if(j.eq.10) ytt=0.07
      call plot(xtt,-ytt,2)
  300 continue
      do 350 i=1,n2+1
      yt=yseg*(i-1)
      call number(-0.38,yt-0.07,0.1,10.0,0.0,-1)
      yp=xy(3)+i-1
      call number(999.,yt-0.04,0.06,yp,0.0,-1)
      if(i.eq.n2+1) go to 350
      do 350 j=2,10
      ytt=yt+yseg*alog10(real(j))
      call plot(0.0,ytt,3)
      xtt=0.04
      if(j.eq.10) xtt=0.07
      call plot(-xtt,ytt,2)
  350 continue
      call symbol(xl/3.0,yl/1.05,0.12,names,0.0,20)
      call symbol(-0.45,yl/2.1,0.14,'AMP',90.0,3)
      names='FREQ (Hz)    '
      if(iaxis.eq.1) names='PERIOD (sec)'
      call symbol(xl/2.5,-0.45,0.14,names,0.0,12)
  360 continue
      xy(5)=real(n1)/xl
      xy(6)=real(n2)/yl
      return
      end
c
c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine sppcal(nn)
      common/dimen/ x(1026),y(1026)
      common/ctrl/  ictrl,size,xy(6)
      do 100 i=1,nn
      x(i)=alog10(x(i))
      y(i)=alog10(y(i))
      if(x(i).lt.xy(1)) x(i)=xy(1)
      if(x(i).gt.xy(2)) x(i)=xy(2)
```

```
      if(y(i).lt.xy(3)) y(i)=xy(3)
      if(y(i).gt.xy(4)) y(i)=xy(4)
  100 continue
      x(nn+1)=xy(1)
      x(nn+2)=xy(5)
      y(nn+1)=xy(3)
      y(nn+2)=xy(6)
      call line(x,y,nn,1,0,0)
      return
      end
***********
```

```
c       program bigfft
c       This performs a fft of length m=2n to yield a
c       2n series by doing only an n length fft
c
c       I/O is accomplished by means of the file 'bigfft.d'.
c       The data in 'bigfft.d' should be:
c               m,dt,df,isign
c               real,imag          or         point1,point2
c                   .                         point3,point4
c                   .                             .
c                   .                             .
c          <total: m/2+1 complex data>      <total: m time data>
c       use  do xxx i=1,m/2+1              do xxx i=1,m,2
c        xxx write(1) data(i)          xxx write(1) time(i),time(i+1)
c       to set up the data file 'bigfft.d' in main program.
c
c       isign=-1 :'forward' FFT, output the lowest  freq first.
c            =-2 :'forward' FFT, output the highest freq first.
c            =+1 :'inverse' FFT, input  the lowest  freq first.
c            =+2 :'inverse' FFT, input  the highest freq first.
c
c       dt,df defined in the fft program four.f are
c       if(isign.lt.0.or.df.eq.0.0) df=1./(nn*dt)
c       if(isign.gt.0.or.dt.eq.0.0) dt=1./(nn*df)
c
c
c       This shows how to use 'bigfft':
c
c       Input:
c              m,dt,df
c              time(i),i=1,m       --time series
c
c       Program:
c
c       open(1,file='bigfft.d',status='new',form='unformatted')
c       rewind 1
c       isign=-1
c       write(1) m,dt,df,isign
c       do 100 i=1,m,2
c100    write(1) time(i),time(i+1)
c       close(1)
c       call system('bigfft',kturn)
c       open(1,file='bigfft.d',status='old',form='unformatted')
c       rewind 1
c       read(1) m,dt,df,isign
c       do 200 i=1,n/2+1
c200    read(1) spec(1,i),spec(2,i)
c
c       Output:
c               m,dt,df,isign
c               spec(i),i=1,m/2+1  -- complex spectrum
c*********
        complex data(4096),x,y,z,twidle,twidel
```

```
      dimension data2(2,4096)
      equivalence (data(1),data2(1,1))
      double precision cos0,cosd,sin0,sind,pp,r12,g12,rr,gg
      open(1,file='bigfft.d',status='old',form='unformatted')
      rewind 1
      read(1) m,dt,df,isign
      n=m/2
      n2=n/2
      np1=n+1
      pp=3.14159265358979d+0/float(n)
      cosd=dcos(pp)
      sind=dsin(pp)
      twidle=cmplx(cosd,sind)
      if(isign.lt.0) go to 1000
      open(2,file='fft.1',status='scratch',form='unformatted')
      rewind 2
      if(isign.ne.+2) go to 200
      do 100 i=np1,1,-1
      read(1) data(i)
  100 continue
      rewind 1
      write(1) m,dt,df,isign
      do 150 i=1,np1
      write(1) data(i)
  150 continue
  200 continue
c     do the work
      do 500 j=1,2
      i=1
      rewind 1
      read(1) m,dt,df,isign
      read(1) x
      if(j.eq.1) then
            data(i)=x
            i=i+1
      endif
      do 300 k=1,n2
      ii=n+2-i
      read(1) y
      read(1) x
      if(j.eq.1) then
            data(i)=x
            data(ii)=conjg(x)
      else
            data(i)=y
            data(ii-1)=conjg(y)
      endif
      i=i+1
  300 continue
      call four(data,n,+1,dt,df)
      dt=0.5*dt
      if(j.ne.1) go to 500
      do 400 i=1,n
```

```
      write(2) data(i)
  400 continue
  500 continue
      rewind 1
      rewind 2
      twidel=cmplx(1.0,0.0)
      do 600 i=1,n
      read(2) x
      z=twidel*data(i)
      y=x+z
      z=x-z
      data(i)=cmplx(real(y),real(z))
      twidel=twidel*twidle
  600 continue
c     now consider data as a real time series of length 2n
c     the order of time values is 1 n+1 2 n+2 3 n+3 ..... n 2n
c     we now have to demultiplex
      rewind 2
      do 700 i=1,n,2
      write(2) data(i),data(i+1)
  700 continue
      rewind 2
      do 800 i=1,n2
      ii=i+n2
      read(2) x,y
      data2(1,i)=real(x)
      data2(2,i)=real(y)
      data2(1,ii)=aimag(x)
      data2(2,ii)=aimag(y)
  800 continue
      rewind 1
      write(1) m,dt,df,isign
      do 900 i=1,n
      write(1) data2(1,i),data2(2,i)
  900 continue
      close(2)
      go to 2000
 1000 continue
c     forward Fourier transform
      do 1100 i=1,n
      read(1) data2(1,i),data2(2,i)
 1100 continue
      call four(data,n,-1,dt,df)
      df=0.5*df
      cos0=1.0d+0
      sin0=0.0d+0
      data2(1,n+1)=data2(1,1)-data2(2,1)
      data2(1,1)=data2(1,1)+data2(2,1)
      data2(2,n+1)=0.0
      data2(2,1)=0.0                n2 = n/2
      do 1200 i=2,n2+1
      pp  =cos0*cosd-sin0*sind
      sin0=sin0*cosd+cos0*sind
```

```
      cosO=pp
      i1=n+2-i
      r1=data2(1,i)
      r2=data2(1,i1)
      g1=data2(2,i)
      g2=data2(2,i1)
      r12=r1-r2
      g12=g1+g2
      rr=cosO*g12-sinO*r12
      gg=sinO*g12+cosO*r12
      gg=-gg
      r12=r1+r2
      g12=g1-g2
      data2(1,i) =r12+rr
      data2(1,i1)=r12-rr
      data2(2,i) =gg+g12
      data2(2,i1)=gg-g12
 1200 continue
      do 1300 i=2,n
      data2(1,i)=0.5*data2(1,i)
      data2(2,i)=0.5*data2(2,i)
 1300 continue
      rewind 1
      write(1) m,dt,df,isign
      do 1400 i=1,np1
      i1=i
      if(isign.eq.-2) i1=n+2-i
      write(1) data(i1)
 1400 continue
 2000 continue
      close(1)
      stop
      end
c
c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c
      subroutine four(data,nn,isign,dt,df)
c     FFT
      dimension data(1)
      n = 2 * nn
      if(isign.eq.-1 .or. df.eq.0.0) df=1.0/(nn*dt)
      if(isign.eq.+1 .or. dt.eq.0.0) dt=1.0/(nn*df)
      j = 1
      do 5 i=1,n,2
      if(i-j)1,2,2
  1   tempr = data(j)
      tempi = data(j+1)
      data(j) = data(i)
      data(j+1)=data(i+1)
      data(i) = tempr
      data(i+1) = tempi
  2   m = n/2
  3   if(j-m) 5,5,4
```

```
 4    j = j-m
      m = m/2
      if(m-2)5,3,3
 5    j=j+m
      mmax = 2
 6    if(mmax-n) 7,10,10
 7    istep= 2 *mmax
      theta = 6.283185307/float(isign*mmax)
      sinth=sin(theta/2.)
      wstpr=-2.*sinth*sinth
      wstpi=sin(theta)
      wr=1.0
      wi=0.0
      do 9 m=1,mmax,2
      do 8 i=m,n,istep
      j=i+mmax
      tempr=wr*data(j)-wi*data(j+1)
      tempi=wr*data(j+1)+wi*data(j)
      data(j)=data(i)-tempr
      data(j+1)=data(i+1)-tempi
      data(i)=data(i)+tempr
 8    data(i+1) = data(i+1)+tempi
      tempr = wr
      wr = wr*wstpr-wi*wstpi + wr
 9    wi = wi*wstpr+tempr*wstpi + wi
      mmax = istep
      go to 6
 10   continue
      if(isign.lt.0) go to 1002
c     frequency to time domain
      do 1001 iiii = 1,n
 1001 data(iiii) = data(iiii) * df
      return
 1002 continue
c     time to frequency domain
      do 1003 iiii = 1,n
 1003 data(iiii) = data(iiii) * dt
      return
      end
**********************
**********************
```