Asymptotic Calculation of the Biot's Waves in Porous Layered Fluid-Saturated media

----

A Thesis

Presented to

the Faculty of the Department of Earth and Atmospheric Sciences

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By

Yangjun (Kevin) Liu

March 2009

Asymptotic Calculation of the Biot's Waves in Porous Layered Fluid-Saturated media

Yangjun (Kevin) Liu

APPROVED:

Dr. Gennady Goloshubin
Advisor

Dr. John Castagna
Advisor

Dr. Chris Liner
Committee member

Dr. Sandhya Devi, Shell
Committee member

Dr. John L. Bear
Dean of the College of Natural Sciences and Mathematics

ACKNOWLEDGEMENTS

A special thank goes to my advisors Dr. Gennady Goloshubin and Dr. John Castagna for their wise leading and knowledgeable demonstrations to me. Without their guidance, I would not be able to finish this project.

I also wish to extend my gratitude to other committee members, Dr. Chris Liner and Dr. Sandhya Devi for their useful comments and suggestions. Dr. Liner showed me how to use Seismic Unix and the art of writing computer programs. Dr. Sandhya Devi suggested learning OASES software for modeling and comparison with my own program.

I want to thank professors in the Department of Earth and Atmospheric Sciences at the University of Houston, especially the Geophysics division, for the excellence courses they taught. I also want to thank my friends Kenny Lew, Shuqin Ma, Jingqiu Huang, etc. for the help and discussions. I wish them good luck in their future endeavors.

Last but not least, I want to thank my parents Yunfu Liu and Lihua Liang for their encouragement and support through this process.

Asymptotic Calculation of the Biot's Waves in Porous Layered Fluid-Saturated media

---

An Abstract of a Thesis

Presented to

the Faculty of the Department of Earth and Atmospheric Sciences

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By

Yangjun (Kevin) Liu

March 2009

ABSTRACT

This work uses the asymptotic analysis of Biot's poroelasticity theory (Goloshubin et al., 2008) to model the seismic response from porous permeable fluid-saturated reservoir. The Thomson-Haskell propagator matrix method is utilized for solving the Fast P wave and Biot's Slow wave propagation through the multi-layered media. We derived the propagator matrix for the normal incidence mode conversion between P wave and Biot's Slow wave and programmed it into a Fortran code. This code calculates the reflectivity and transmitivity series of a fluid zone and obtains the influence of the Slow P wave on the seismic signal.

Our results show that for a reservoir with homogeneous fluid saturation, Slow P wave effect is negligible. If the rock is inhomogeneous in either fluid saturation or permeability, a significant Slow P wave effect can be observed. The Slow P wave effect is very sensitive to frequency and has a strong similarity with the observed low frequency shadows. It is highly possible that the low frequency shadows frequently observed under gas reservoirs are induced by the fluid flow in the reservoir and the propagation of the Slow P wave.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: Introduction

Biot's poroelasticity theory (1956a, b) predicts movements of the pore fluid relative to the skeleton as seismic waves propagate through the reservoir. This phenomenon opens an opportunity for investigation of fluid properties of the hydrocarbon-saturated reservoirs from seismic amplitude. According to Biot, a compressional P wave in a fluid-saturated porous medium is a superposition of slow and fast waves. These two waves are always coupled and coexist with each other, so that in a fluid-saturated porous permeable rock, three types of waves exist: classical P wave, Shear wave, and Biot's Slow wave, as shown in Figure1. The left figure shows wave propagation at a pure elastic boundary, where only P wave and shear wave exist, the right figure shows wave propagation at a porous permeable elastic boundary, where P wave, Shear wave and Biot's Slow wave would exist.

One application of Biot's theory is the study of Dutta & Ode (1983). They used Biot's model to calculate the seismic reflections from a gas-water boundary in a porous sand reservoir. Velocity, attenuation, and angular-dependent reflection and transmission at both gas and water layers for the frequency range 0 ~ 10^5 Hz were obtained. They concluded that in a fluid-saturated porous rock, loss of seismic energy is mainly due to the mode conversion to Biot's Slow wave and they are proportional to $f^{1/2}$, where f is the frequency. There is about 2.5 percent of energy loss due to mode conversion to Biot's Slow wave at 100 Hz. Thus, They suggested that in fluid saturated rock, effects due to Slow wave should be taken into account.

Dutta & Ode (1979a, b) and Dutta & Seriff (1979) studied the attenuation of seismic wave in a fluid-saturated porous rock with partial gas saturation (White, 1975) using Biot's theory and modified White's model, respectively. Both theories conclude in good agreement with each other. They also pointed out that the energy dissipation for their model is mainly due to the relative fluid flow from Biot's Slow wave. Three different geometries of gas-filled zones are analyzed and compared in this study. They are: sphere model, shell model, and layer model. While in terms of magnitude of the attenuation, all three models behave similarly.

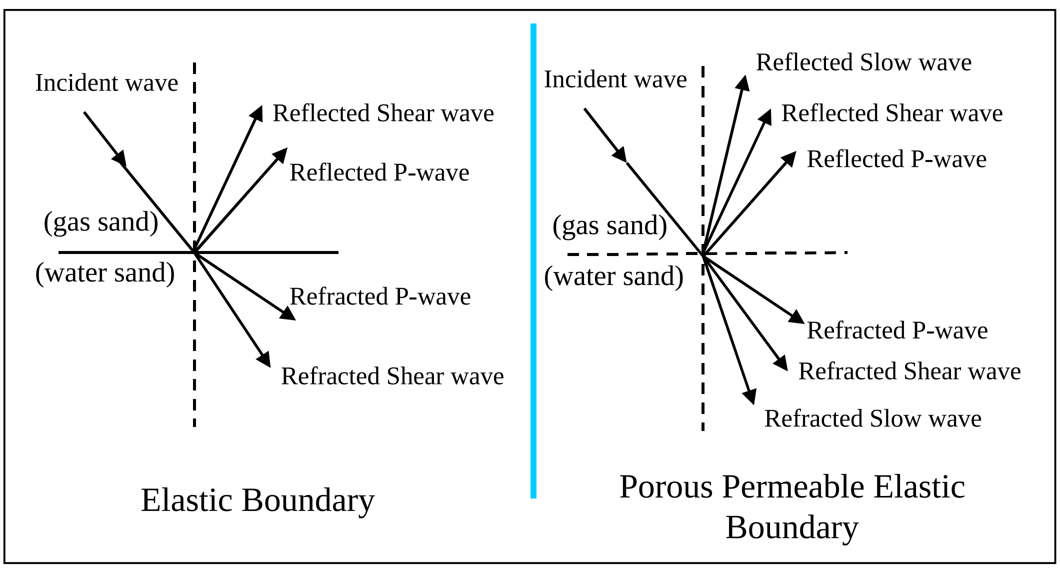

Figure 1. Reflection and refraction of a wave from elastic boundary and from a porous permeable elastic boundary.

Carcione et al. (2003) utilized a poroelastic modeling algorithm to compute wave propagation in White's spherical gas pockets model. Their results also confirm that the conversion of fast P wave into Biot's Slow wave is the main mechanism of attenuation for a partial gas saturated, brine-filled porous rock.

Our work presented here utilizes a propagator matrix method to calculated full-wave reflectivity series from a fluid-saturated porous permeable rock that composed of layered media, reflection and transmission coefficients through medium boundary are provided by asymptotic solutions of Biot's theory (Silin & Goloshubin, 2008). In this case, only normal incident scenarios are calculated, given the full solution of asymptotic analysis of Biot at normal incidence. Our result shows that Slow wave effects may appear in a seismic section as some real P wave reflectors. Its relative amplitude strongly depends on frequency and fluid type. When partial gas saturation exists in a fluid-saturated porous permeable rock, Slow wave effect is significantly enhanced. Agree with previous studies, conversion from P wave to Slow wave is the main mechanism of energy loss. Resonance due to recursive reflection of Slow waves among layers is possible, which would strongly enhance the seismic amplitude. This occurs only at very small sample rate such as 0.1 ms, since Slow wave effect can be resolved before all attenuated. If the sample rate are taken to be 1 ms ~ 4 ms, resonance due to Slow wave is less likely to occur. Slow wave in this study only refers to Biot's Slow wave, classical P wave is also called to Fast P wave.

CHAPTER 2: Asymptotic Calculation

2.1 Fast P wave as incident wave

Present work applies asymptotic formulas of Silin and Goloshubin (2008) to calculate reservoir models that constitute thin layers of gas and water layers. First, I study the seismic reflection from a single gas-water contact. In this situation, only the normal incident P wave is treated as the input wave, the output waves include both the reflected and transmitted fast and slow waves. They are denoted by R^{FF}, T^{FF}, R^{FS}, T^{FS}, and can be written in the asymptotic form as follows:

$$R^{FF} = \frac{Z_1^F - Z_2^F}{Z_1^F - Z_2^F} + R_1^{FF} \frac{1+i}{\sqrt{2}} \sqrt{|\varepsilon|} + \ldots$$

$$T^{FF} = 1 + \frac{Z_1^F - Z_2^F}{Z_1^F + Z_2^F} + T_1^{FF} \frac{1+i}{\sqrt{2}} \sqrt{|\varepsilon|} + \ldots$$

$$R^{FS} = R_1^{FS} \frac{1+i}{\sqrt{2}} \sqrt{|\varepsilon|}; \text{ and } T^{FS} = T_1^{FS} \frac{1+i}{\sqrt{2}} \sqrt{|\varepsilon|};$$

where, Z_1 and Z_2 are the modified acoustic impedances of medium 1 and 2:

$$Z = \frac{M}{\upsilon_b} \sqrt{\frac{\gamma_\beta + (\gamma_M)^2}{\gamma_\beta}};$$

Here, $M = K + \frac{4}{3}\mu$ is the plane wave modulus from the dry rock bulk modulus K and shear modulus μ, $\upsilon_b = \sqrt{\frac{M}{\rho_b}}$, (ρ_b is the bulk density) and γ_β, γ_M are given by:

$$\gamma_\beta = M\left(\beta_f \phi + \frac{1-\phi}{K_{fg}}\right) \text{ and } \gamma_M = 1 - \frac{(1-\phi)K}{K_{sg}}$$

where, $\beta_f = \frac{1}{K_f}$ is the fluid compressibility.

Also, first order refelction and transmission coefficients R_1^{FF} and T_1^{FF} have the forms:

$$R_1^{FF} = \frac{Z_2(T_1^{FS} - R_1^{FS})}{Z_1 + Z_2}; \text{ and } T_1^{FF} = \frac{Z_1(R_1^{FS} - T_1^{FS})}{Z_1 + Z_2}.$$

We assume i = 1 is the medium above the boundary, i = 2 is the medium below the boundary. Note that γ_β, γ_M are dimensionless parameters. A and D are also dimensionless parameters given by:

$$A = \left[\frac{\gamma_{M1}}{(\gamma_{M1})^2 + \gamma_{\beta 1}} - \frac{\gamma_{M2}}{(\gamma_{M2})^2 + \gamma_{\beta 2}}\right]\frac{2Z_1Z_2}{Z_1 + Z_2};$$

$$D = \frac{Z_1Z_2\sqrt{\gamma_{\beta 1}\gamma_{\beta 2}}}{\gamma_{M1}\gamma_{M2}}\left[\frac{1}{\sqrt{\gamma_\kappa}}\frac{1}{\gamma_{\rho 2}}\frac{\upsilon_{b1}}{M_1}\sqrt{(\gamma_{M1})^2 + \gamma_{\beta 1}} + \frac{1}{\gamma_{\rho 1}}\frac{\upsilon_{b2}}{M_2}\sqrt{(\gamma_{M2})^2 + \gamma_{\beta 2}}\right]$$

where, γ_κ and γ_ρ are:

$$\gamma_\kappa = \frac{\varepsilon_2}{\varepsilon_1} \text{ and } \gamma_\rho = \frac{\rho_b}{\rho_f}$$

ρ_b is the bulk density and ρ_f is the density of the pore fluid.

Finally, K_{sg} and K_{fg} are given by:

$$K_{sg} = \frac{K_g}{1-\phi} \text{ and } K_{fg} = \frac{K_g}{1-\frac{K}{K_g}}$$

K_g is the bulk modulus of solid grain, K is the dry rock bulk modulus, and ϕ is porosity.

The asymptotic solutions provide approximations to Biot's theory, however, more explicit descriptions on the rock and fluid properties are obtained in asymptotic formulas. Table 1 summarizes the input parameters needed in this calculation. Where, K_g and ρ_g are the bulk modulus and density of the solid grain; K_{dry} and μ_{dry} are the dry rock bulk modulus and shear modulus; ϕ and κ are porosity and permeability of the rock; K_f, ρ_f, and η_f are the bulk modulus, density and viscosity of the filling fluid, respectively. Note that all these input parameters are no more than the ones used in doing a fluid substitution with Gassmann's equation. They can be easily acquired from log data. Therefore, wider applications of Biot's theory may be obtained through these asymptotic solutions. (Goloshubin et al., 2008).

Table 1. Input properties for asymptotic Biot's calculation.

Input prop.	Grain bulk mod.	Grain dens.	Dry rock bulk mod.	Dry rock shear mod.	Poro.	Perm.	Fluid bulk mod.	Fluid dens.	Fluid visco.
Symbol	K_g	ρ_g	K_{dry}	μ_{dry}	ϕ	κ	K_f	ρ_f	η_f

Further more, velocity (m/s) and attenuation coefficients in units of (1/m) for Fast and Slow waves can be calculated from:

$$V^F = v_b\sqrt{1+\frac{\gamma_M^2}{\gamma_\beta}} + ...;$$

$$V^S = v_f\sqrt{\frac{2|\varepsilon|}{\gamma_\beta + \gamma_M^2}} + ...;$$

$$a^F = \frac{\omega}{v_b}\sqrt{\frac{\gamma_\beta}{\gamma_\beta + \gamma_M^2}}\frac{\varsigma_1^F}{2\varsigma_0^F}|\varepsilon| + ...;$$

$$a^S = \frac{\omega}{v_f}\sqrt{\frac{\gamma_\beta + \gamma_M^2}{2|\varepsilon|}} + ...;$$

where, $v_f = \sqrt{\frac{M}{\rho_f}}$, and $\varsigma_0^F, \varsigma_1^F$ are given by:

$$\varsigma_0^F = \frac{\gamma_M^2 + \gamma_\beta}{\gamma_\beta\gamma_\rho}, \text{ and } \varsigma_1^F = \frac{1}{\gamma_\beta(\gamma_M^2 + \gamma_\beta)}\left(\frac{\gamma_M^2 + \gamma_\beta}{\gamma_\rho} - \gamma_M\right)^2$$

2.2 Biot's Slow wave as incident wave

Asymptotic solution obtains the reflection and transmission coefficients from incident Slow wave converts to both Fast P wave and Slow waves as follows:

$$R^{SS} = \frac{-\chi_{01}^S\frac{1}{\sqrt{\gamma_\kappa}}M_2k_{02}^S\xi_{02}^S + \chi_{02}^S M_1 k_{01}^S\xi_{01}^S}{\chi_{01}^S\frac{1}{\sqrt{\gamma_\kappa}}M_2k_{02}^S\xi_{02}^S + \chi_{02}^S M_1 k_{01}^S\xi_{01}^S};$$

$$T^{SS} = \frac{\chi_{02}^{S} \frac{1}{\sqrt{\gamma_{\kappa}}} M_2 k_{02}^{S} \xi_{02}^{S} + \chi_{01}^{S} M_1 k_{01}^{S} \xi_{01}^{S}}{\chi_{01}^{S} \frac{1}{\sqrt{\gamma_{\kappa}}} M_2 k_{02}^{S} \xi_{02}^{S} + \chi_{02}^{S} M_1 k_{01}^{S} \xi_{01}^{S}};$$

$$R^{SF} = \frac{Z_2(-1 - R^{SS} + T^{SS})}{Z_1 + Z_2};$$

$$T^{SF} = \frac{-Z_1(-1 - R^{SS} + T^{SS})}{Z_1 + Z_2},$$

where, $k_0^S = \frac{1}{v_f}\sqrt{\gamma_\beta + \gamma_M^2}$, $\chi_0^S = \frac{\gamma_M^2 + \gamma_\beta}{\gamma_M}$ and $\xi_0^S = -\frac{1}{\gamma_M}$.

2.3 Asymptotic calculation on gas-water contact

In the first calculation using asymptotic formulas of Biot, we use the gas-water contact model, i.e., a gas sand overlying a water sand, while the rock properties for both the gas zone and water zone are kept same. For comparison purpose, all the properties are taken from Dutta & Ode (1983) for unconsolidated, Texas Gulf coast sand at depth of about 1500 m, as shown in Table 2.

Table 2. Rock and fluid properties of gas and water saturated sand.

Input prop.	Grain bulk mod.	Grain dens.	Dry rock bulk mod.	Dry rock shear mod.	Poro.	Perm.	Fluid bulk mod.	Fluid dens.	Fluid visco.
Symbol	Kg (Gpa)	ρ_g (g/cc)	K_{dry} (Gpa)	μ_{dry} (Gpa)	ϕ	κ (darcy)	K_f (Gpa)	ρ_f (g/cc)	η_f (cp)
Gas zone	35	2.65	1.7	1.855	0.3	1	0.022	0.1	0.015
Water zone	35	2.65	1.7	1.855	0.3	1	2.4	1	1

2.3.1 Velocity versus frequency

The velocity versus frequency plot are displayed in Figure 2 for both P wave and Biot's Slow wave in gas layer and water layer, an extrapolated curve from Dutta & Ode is also plotted as a comparison to the Slow wave velocity from present calculation. P wave velocity for both water and gas layers stay as constant through out the 10 ~ 10^5 Hz frequency range, while a strong velocity dispersion is calculated for the Slow wave, both in water and gas layers. In the low seismic frequency range (10 ~ 100 Hz), the asymptotic calculations show good agreement with exact Biot's calculation of Dutta and Ode (1983).

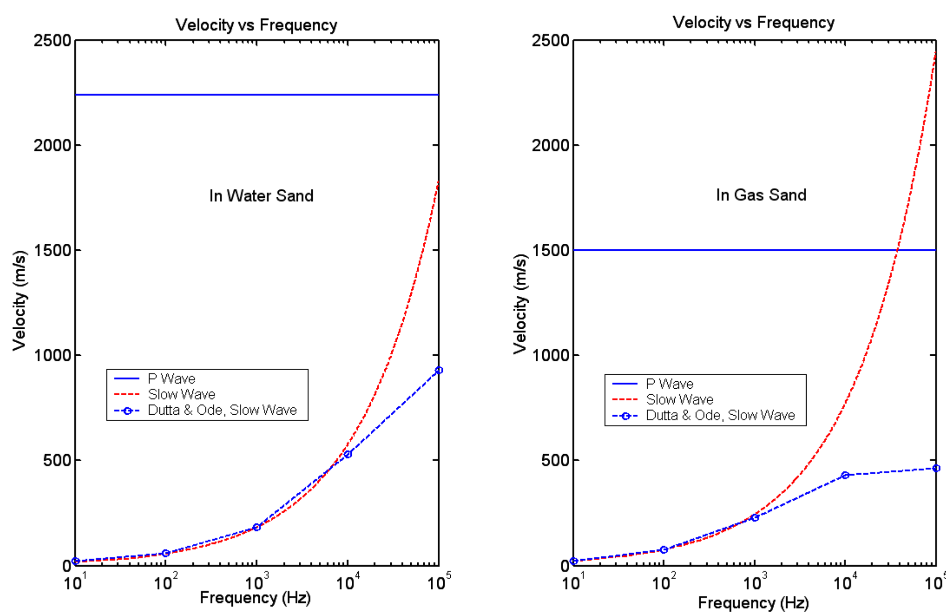

Figure 2. Velocity dispersion of the P wave and Biot's Slow wave in gas and water saturated porous rocks.

2.3.2 Attenuation versus frequency

The attenuation of the P wave and Biot's Slow wave in water and gas saturated rocks are displayed in Figure 3. The attenuation of Biot's Slow wave is about 10^5 times greater than that for the Fast P wave at the seismic frequency range. By comparing the curves of asymptotic calculations (blue) and Dutta & Ode's exact Biot's calculations (red), strong similarities are found for both gas and water saturated rocks at low frequencies below 10^3 Hz. High frequencies, on the other hand, proved to have a low agreement between these two calculations. Attenuation of P wave is tiny with respect to the attenuation of Slow wave, however in a fluid saturated zone, the attenuation of P wave is enhanced by a mechanism of conversion to Biot's Slow wave. Studies of Dutta & Ode (1979a, b) and Dutta & Seriff (1979) have showed this trend. Asymptotic calculations provide similar results with their studies in terms of attenuation.

Lists of reflection and transmission coefficients for up and down going waves at 22 Hz are summarized in Table 3. Reflection coefficient of Fast P wave to Slow wave (Rfs) in water sand (equals to 0.025977), is much larger than the reflection coefficient (Rfs) in gas sand (equals to -0.000105); transmission coefficient of Fast P wave to Slow wave (Tfs) from gas sand to water sand (equals to -0.003897), is much larger than the transmission coefficient (Tfs) from water sand to gas sand (equals to 0.000697). This suggests that the amplitude of Slow P wave always increases going from gas zone to water zone, and decreases going from water zone to gas zone. Slow P wave needs relatively incompressible fluid to support it.

Table 3. Results of asymptotic calculations at 22 Hz on reflection and transmission coefficients for up and down going waves at the gas-water contact shows in Table 2.

Wave Type	P incidence Down going	Slow incidence Down going	P incidence Up going	Slow incidence Up going
Reflection to P wave	-0.263028	0.453893	0.251292	-0.444078
Reflection to Slow wave	-0.000105	-0.960673	0.025977	0.960673
Transmission to P wave	0.740764	-0.266208	1.276572	0.757167
Transmission to Slow wave	-0.003897	0.759428	0.000697	0.759428

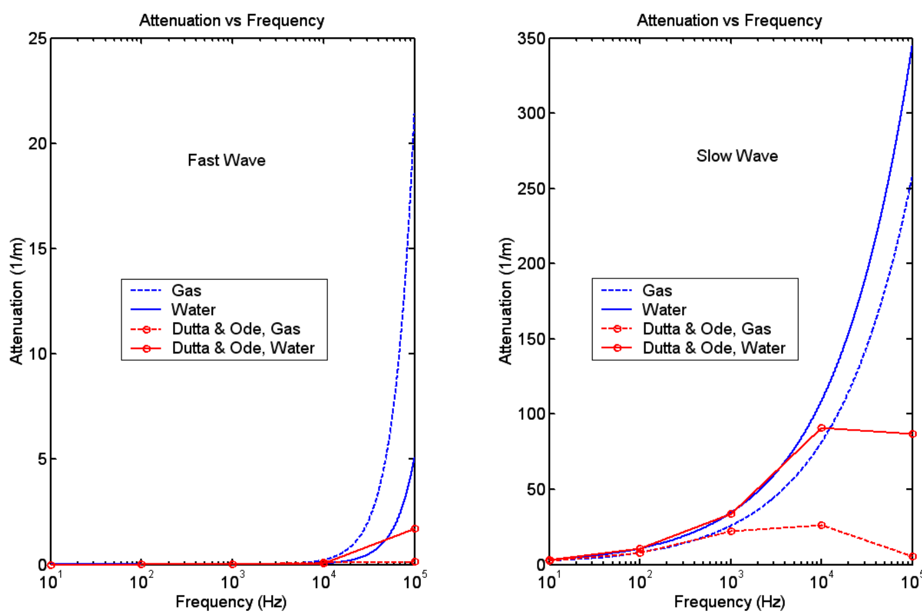

Figure 3. Attenuation coefficients vs. Frequency and comparison with Dutta & Ode's calculations.

According to Table 3, the largest value of the coefficient for conversion from Fast P wave to Slow P wave is the reflection from water sand to water sand (equals to 0.02597). This amount is similar to Dutta & Ode's (1983) study, that they observed 2.5% of energy loss due to mod conversion from P wave to Slow wave. Since Slow P wave attenuates fast, most of the Slow wave energy would dissipate through fluid oscillations.

Therefore, from the comparisons of velocity and attenuation, asymptotic calculation provides similar results with exact Biot's solution at seismic frequency range. Since asymptotic calculation simplifies the algorithm of Biot's theory and relates the model more explicitly with rock and fluid properties, it can be more practically used as a tool for seismic inversion, simulation and reservoir characterization, etc.

2.3.3 Reflections versus frequency

The reflection and transmission coefficients for Fast P wave as an incident wave entering from gas zone onto a gas-water contact are displayed in Figure 4. Both the conversion to Fast P wave and Biot's Slow wave are plotted. Reflection (Rff) and transmission coefficient (Tff) are compared with the results from classical elastic calculations. We can see a significant frequency dependency of the Rff and Tff, while the classical Rff and Tff are not dependent on frequency. The polarity in our reflection coefficient is from European convention. A negative reflection coefficient exists for wave goes from low impedance rock to high impedance rock.

Phase angle in degrees for Rff, Tff, Rfs, and Tfs are also plotted as a function of frequency. As frequency increases, phase angle increases for Rff and Tff. Rff phase increases faster than that for Tff phase, but generally small phase angles being less than 5 degrees are observed for both Rff and Tff within seismic frequency range. Since Rfs and Tfs only have first order term with respect to frequency in asymptotic formulas, their phase stays as a constant throughout all frequency range.

In Table 4, the 100 Hz and 10 KHz results from asymptotic calculations and from Dutta & Ode's calculations at normal incidence are summarized. Correct the polarity difference between asymptotic calculation and Dutta & Ode's calculation, their values for Rff and Tff are very close at 100 Hz. At 10 KHz, both values for Rff and Tff are very different. According to the comparison for velocity and attenuation, asymptotic calculation will no longer match with exact Biot's calculations for high frequency above 1 KHz, the relative big deviation at 10 KHz can be expected. This comparison further proves that Asymptotic calculation matches with the exact Biot's calculation at seismic frequency domain, thus it can be used for modeling and analysis to seismic data.

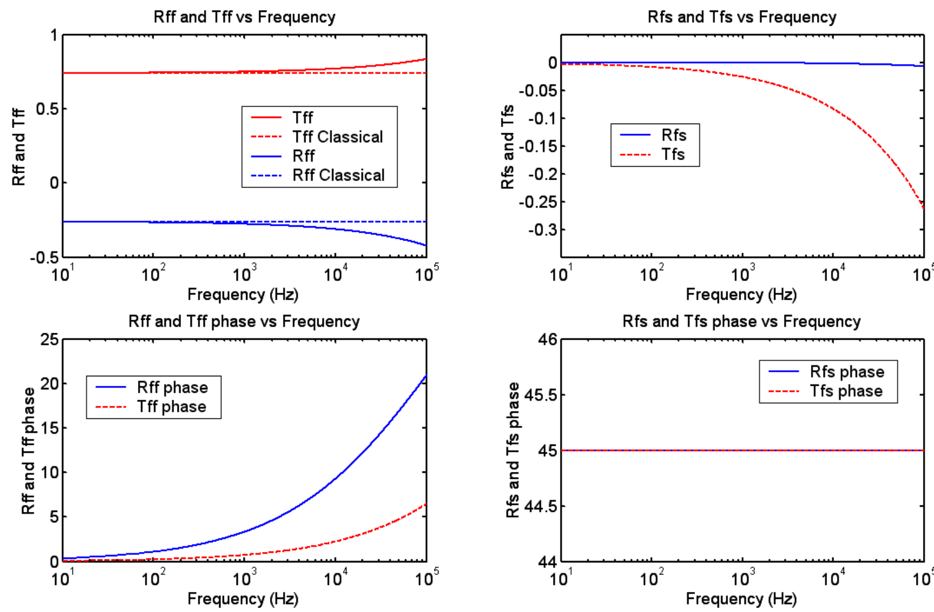

Figure 4. Reflection and transmission coefficients and phase angle for the wave propagation at the gas-water contact in Table 2. Incident wave is from gas sand.

Table 4. Comparison between asymptotic calculations and Dutta & Ode's calculations on the reflection and transmission coefficients (Rff and Tff) for the gas-water contact in Table 2, normal incidence.

Frequency	100 Hz		10 KHz	
Type	Rff	Tff	Rff	Tff
Asymptotic	0.241	0.742	0.061	0.769
Dutta & Ode	0.246	0.744	0.175	0.690

CHAPTER 3: Modeling Multi-layered Media

3.1 Propagator matrix method

In this section, we derive the classical propagator matrix method to obtain 1-D reflectivity and transmitivity at a single frequency for multi-layered fluid-saturated porous permeable media. Calculation is only for fluid zone and assumption is that both source and receiver are located at the half space. Robinson (1967) provided a good demonstration on applying propagator matrix method to solve P wave propagation in layered media with z-transform. More general description of this method is presented in Aki & Richards (1980). Here we modified the algorithm to include the mod conversion between P wave and Biot's Slow wave at normal incidence.

Figure 5 shows a boundary condition of the Fast and Slow P-waves traveling vertically at an arbitrary layer j. The ray paths are drawn with time displacement along the horizontal axis, which helps in displaying the vertically traveling rays through time axis. Indeed, all the ray paths are perpendicular to the horizontal plane. In our treatment, each layer is assumed to have same one-way travel time for P wave propagating through one layer. And this one-way travel time is taken to be one unit of time for Fast P wave; and τ unit of time for Slow P wave. The τ value changes from layer to layer, depends on the ratio of the velocities of Fast P wave *vfast* and Slow P wave *vslow*. Namely, for an arbitrary layer j, we have:

$$\tau_j = \frac{vfast_j}{vslow_j}.$$

In Figure 5, we denote the downgoing Fast P wave at the top of layer j by $d_j(t)$, and the downgoing Slow wave at the top of layer j by $d_j'(t)$. Then, if there is no attenuation in layer j, the downgoing wave at the bottom of layer j will be the same waveform delayed by the one-way travel time for the layer j (which is defined as one time unit for Fast P wave, and τ time unit for Slow P wave); hence the downgoing Fast P wave at the bottom of layer j is $d_j(t-1)$, the downgoing Slow P wave at the bottom of layer j is $d_j'(t-\tau)$. Similarly, we denote the upgoing Fast P wave at the top of layer j by $u_j(t)$, and the upgoing Slow P wave at the top of layer j by $u_j'(t)$. Then the upgoing Fast P wave at the bottom of layer j is the same wave but advanced by one time unit, i.e., $u_j(t+1)$, and the upgoing Slow P wave at the bottom of layer j is $u_j'(t+\tau)$.

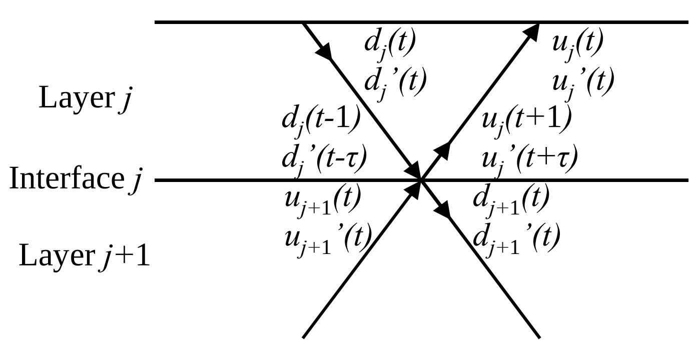

Figure 5. Schematic plot of wave propagation through layer j at normal incidence. The horizontal displacement corresponds to time.

If attenuation of waves in layer j is considered, then, the downgoing Fast P wave at the bottom of layer j becomes $A_f \cdot d_j(t-1)$, where A_f is the attenuation of P wave. The downgoing Slow P wave at the bottom of layer j becomes $A_s \cdot d_j'(t-1)$, where A_s is the attenuation of Slow wave. Similarly, the upgoing Fast P wave at the bottom of layer j becomes $A_f^{-1} \cdot u_j(t+1)$, and the upgoing Slow P wave at the bottom of layer j becomes $A_s^{-1} \cdot u_j'(t+\tau)$. The expressions of A_f and A_s are shown below:

$$A_f = \exp(-\alpha_j^f h_j) = \exp(-\alpha_j^f (vfast_j \cdot dt));$$

$$A_s = \exp(-\alpha_j^s h_j) = \exp(-\alpha_j^s (vfast_j \cdot dt)),$$

where, dt is the unit of time or sample time; α_j^s is the attenuation coefficient of Slow P wave and α_j^f is the attenuation coefficient of Fast P wave in units of (m^{-1}).

According to the boundary condition in Figure 5, we can obtain the relationships between all waveforms at interface j with the corresponding reflection and transmission coefficients. We use *rff* to represent the reflection coefficient of Fast P wave to Fast P wave while the incident P wave is downgoing, and *rffup* to represent the reflection coefficient of Fast P wave to Fast P wave while the incident Fast P wave is upgoing. Similar denotations are used for other reflection and transmission coefficients.

The wave $d_{j+1}(t)$ is made up of four parts, i.e., the parts due to transmitted portion of $d_j(t-1)$ and $d_j'(t-\tau)$, and the parts due to the reflected portion of $u_{j+1}(t)$ and $u_{j+1}'(t)$. Thus it gives the equation:

$$d_{j+1}(t) = tff_j \cdot A_f \cdot d_j(t-1) + tsf_j \cdot A_s \cdot d_j'(t-\tau_j) + rffup_j \cdot u_{j+1}(t) + rsfup_j \cdot u_{j+1}'(t),$$

Similarly the wave $d_{j+1}'(t)$ is made up of four parts, i.e., the transmitted portion of $d_j(t\text{-}1)$ and $d_j'(t\text{-}\tau)$, and the reflected portion of $u_{j+1}(t)$ and $u_{j+1}'(t)$. Thus we have the equation:

$$d'_{j+1}(t) = tfs_j \cdot A_f \cdot d_j(t-1) + tss_j \cdot A_s \cdot d'_j(t-\tau_j) + rfsup_j \cdot u_{j+1}(t) + rssup_j \cdot u'_{j+1}(t).$$

Similarly the waves $u_j(t+1)$ and $u_j'(t+\tau)$ are made up of four parts, i.e., the reflected portion of $d_j(t\text{-}1)$ and $d_j'(t\text{-}\tau)$, and the transmitted portion of $u_{j+1}(t)$ and $u_{j+1}'(t)$. Thus we have the equations:

$$A_f^{-1} \cdot u_j(t+1) = rff_j \cdot A_f \cdot d_j(t-1) + rsf_j \cdot A_s \cdot d'_j(t-\tau_j) + tffup_j \cdot u_{j+1}(t) + tsfup_j \cdot u'_{j+1}(t);$$

$$A_s^{-1} \cdot u'_j(t+\tau_j) = rfs_j \cdot A_f \cdot d_j(t-1) + rss_j \cdot A_s \cdot d'_j(t-\tau_j) + tfsup_j \cdot u_{j+1}(t) + tssup_j \cdot u'_{j+1}(t).$$

We define $D_j(s)$ as the Laplace transform of $d_j(t)$, i.e.,

$$D_j(s) = \int_0^\infty d_j(t)e^{-st}dt \cdot$$

$D_j(s)$ represents all the downgoing waveforms at the top of layer j.

The Laplace transform of $d_j(t\text{-}1)$ is

$$\int_0^\infty d_j(t-1)e^{-st}dt = \int_0^\infty d_j(t_1)e^{-s(t_1+1)}dt_1 \ ,$$

If we define $z = e^{-s}$, then,

$$\int_0^\infty d_j(t-1)e^{-st}dt = z^1 D_j(s).$$

$z^1 D_j(s)$ represents all the downgoing waveforms at the bottom of layer j.

Thus, in Laplace transform, the waveform at the bottom of layer j differs with the one at the top of layer j by a multiplication of z^1. We can also view $D_j(s)$ as a polynomial of z, then, the multiplication of $D_j(s)$ by z^1 corresponds to a shift of the polynomial of $D_j(s)$ by one step to the right. Also the multiplication of two waveforms in Laplace transform is equivalent to a convolution of their polynomial coefficients in z. These characters are very useful in doing subsequent polynomial operations.

Similarly, we can find the Laplace transforms for other waves in above equations and construct four new equations in Laplace transforms, namely:

$$D_{j+1}(s) = tff_j \cdot A_f \cdot z^1 D_j(s) + tsf_j \cdot A_s \cdot z^{\tau_j} D'_j(s) + rffup_j \cdot U_{j+1}(s) + rsfup_j \cdot U'_{j+1}(s);$$

$$D'_{j+1}(s) = tfs_j \cdot A_f \cdot z^1 D_j(s) + tss_j \cdot A_s \cdot z^{\tau_j} D'_j(s) + rfsup_j \cdot U_{j+1}(s) + rssup_j \cdot U'_{j+1}(s);$$

$$A_f^{-1} \cdot z^{-1} U_j(s) = rff_j \cdot A_f \cdot z^1 D_j(s) + rsf_j \cdot A_s \cdot z^{\tau_j} D'_j(s) + tffup_j \cdot U_{j+1}(s) + tsfup_j \cdot U'_{j+1}(s);$$

$$A_s^{-1} \cdot z^{-\tau_j} U'_j(s) = rfs_j \cdot A_f \cdot z^1 D_j(s) + rss_j \cdot A_s \cdot z^{\tau_j} D'_j(s) + tfsup_j \cdot U_{j+1}(s) + tssup_j \cdot U'_{j+1}(s).$$

Rearrangement of the above equations leads to the following four equations:

$$D_{j+1}(s) = tff_j \cdot A_f \cdot z^1 D_j(s) + tsf_j \cdot A_s \cdot z^{\tau_j} D'_j(s) + rffup_j \cdot U_{j+1}(s) + rsfup_j \cdot U'_{j+1}(s) \qquad (1)$$

$$D'_{j+1}(s) = tfs_j \cdot A_f \cdot z^1 D_j(s) + tss_j \cdot A_s \cdot z^{\tau_j} D'_j(s) + rfsup_j \cdot U_{j+1}(s) + rssup_j \cdot U'_{j+1}(s) \qquad (2)$$

$$U_{j+1}(s) = \frac{-rff_j}{tffup_j} \cdot A_f \cdot z^1 D_j(s) + \frac{-rsf_j}{tffup_j} \cdot A_s \cdot z^{\tau_j} D'_j(s) + \frac{-tsfup_j}{tffup_j} \cdot U'_{j+1}(s) + \frac{1}{tffup_j} \cdot A_f^{-1} \cdot z^{-1} U_j(s) \qquad (3)$$

$$U'_{j+1}(s) = \frac{-rfs_j}{tssup_j} \cdot A_f \cdot z^1 D_j(s) + \frac{-rss_j}{tssup_j} \cdot A_s \cdot z^{\tau_j} D'_j(s) + \frac{-tfsup_j}{tssup_j} \cdot U_{j+1}(s) + \frac{1}{tssup_j} \cdot A_s^{-1} \cdot z^{-\tau_j} U'_j(s) \qquad (4)$$

Substituting equation (4) into equation (3) leads to:

$$\begin{aligned} U_{j+1}(s) = {} & \frac{-tssup_j \cdot rff_j + tsfup_j \cdot rfs_j}{tffup_j \cdot tssup_j - tsfup_j \cdot tfsup_j} \cdot A_f \cdot z^1 D_j(s) + \frac{-tssup_j \cdot rsf_j + tsfup_j \cdot rss_j}{tffup_j \cdot tssup_j - tsfup_j \cdot tfsup_j} \cdot A_s \cdot z^{\tau_j} D'_j(s) + \\ & \frac{tssup_j}{tffup_j \cdot tssup_j - tsfup_j \cdot tfsup_j} \cdot A_f^{-1} \cdot z^{-1} U_j(s) + \frac{-tsfup_j}{tffup_j \cdot tssup_j - tsfup_j \cdot tfsup_j} \cdot A_s^{-1} \cdot z^{-\tau_j} U'_j(s) \end{aligned} \qquad (5)$$

Substituting equation (5) into equation (4) leads to:

$$U'_{j+1}(s)=\frac{-rfs_j\cdot tffup_j+rff_j\cdot tfsup_j}{tffup_j\cdot tssup_j-tsfup_j\cdot tfsup_j}\cdot A_f\cdot z^1 D_j(s)+\frac{-rss_j\cdot tffup_j+rsf_j\cdot tfsup_j}{tffup_j\cdot tssup_j-tsfup_j\cdot tfsup_j}\cdot A_s\cdot z^{\tau_j}D'_j(s)+ \frac{-tfsup_j}{tffup_j\cdot tssup_j-tsfup_j\cdot tfsup_j}\cdot A_f^{-1}\cdot z^{-1}U_j(s)+\frac{tffup_j}{tffup_j\cdot tssup_j-tsfup_j\cdot tfsup_j}\cdot A_s^{-1}\cdot z^{-\tau_j}U'_j(s) \quad (6)$$

Substituting equation (5) and equation (6) into equation (1) leads to:

$$\begin{aligned}D_{j+1}(s)=&\left[\frac{tff_j\cdot(tffup_j\cdot tssup_j-tsfup_j\cdot tfsup_j)+rffup_j\cdot(-tssup_j\cdot rff_j+tsfup_j\cdot rfs_j)}{tffup_j\cdot tssup_j-tsfup_j\cdot tfsup_j}+\right.\\&\left.\frac{rsfup_j\cdot(-rfs_j\cdot tffup_j+rff_j\cdot tfsup_j)}{tffup_j\cdot tssup_j-tsfup_j\cdot tfsup_j}\right]\cdot A_f\cdot z^1 D_j(s)+\\&\left[\frac{tsf_j\cdot(tffup_j\cdot tssup_j-tsfup_j\cdot tfsup_j)+rffup_j\cdot(-tssup_j\cdot rsf_j+tsfup_j\cdot rss_j)}{tffup_j\cdot tssup_j-tsfup_j\cdot tfsup_j}+\right.\\&\left.\frac{rsfup_j\cdot(-rss_j\cdot tffup_j+rsf_j\cdot tfsup_j)}{tffup_j\cdot tssup_j-tsfup_j\cdot tfsup_j}\right]\cdot A_s\cdot z^{\tau_j}D'_j(s)+\\&\frac{rffup_j\cdot tssup_j-rsfup_j\cdot tfsup_j}{tffup_j\cdot tssup_j-tsfup_j\cdot tfsup_j}\cdot A_f^{-1}\cdot z^{-1}U_j(s)+\frac{-rffup_j\cdot tsfup_j+rsfup_j\cdot tffup_j}{tffup_j\cdot tssup_j-tsfup_j\cdot tfsup_j}\cdot A_s^{-1}\cdot z^{-\tau_j}U'_j(s)\end{aligned} \quad (7)$$

Substituting equation (5) and equation (6) into equation (2) leads to:

$$\begin{aligned}D'_{j+1}(s)=&\left[\frac{tfs_j\cdot(tffup_j\cdot tssup_j-tsfup_j\cdot tfsup_j)+rfsup_j\cdot(-tssup_j\cdot rff_j+tsfup_j\cdot rfs_j)}{tffup_j\cdot tssup_j-tsfup_j\cdot tfsup_j}+\right.\\&\left.\frac{rssup_j\cdot(-rfs_j\cdot tffup_j+rff_j\cdot tfsup_j)}{tffup_j\cdot tssup_j-tsfup_j\cdot tfsup_j}\right]\cdot A_f\cdot z^1 D_j(s)+\\&\left[\frac{tss_j\cdot(tffup_j\cdot tssup_j-tsfup_j\cdot tfsup_j)+rfsup_j\cdot(-tssup_j\cdot rsf_j+tsfup_j\cdot rss_j)}{tffup_j\cdot tssup_j-tsfup_j\cdot tfsup_j}+\right.\\&\left.\frac{rssup_j\cdot(-rss_j\cdot tffup_j+rsf_j\cdot tfsup_j)}{tffup_j\cdot tssup_j-tsfup_j\cdot tfsup_j}\right]\cdot A_s\cdot z^{\tau_j}D'_j(s)+\\&\frac{rfsup_j\cdot tssup_j-rssup_j\cdot tfsup_j}{tffup_j\cdot tssup_j-tsfup_j\cdot tfsup_j}\cdot A_f^{-1}\cdot z^{-1}U_j(s)+\frac{-rfsup_j\cdot tsfup_j+rssup_j\cdot tffup_j}{tffup_j\cdot tssup_j-tsfup_j\cdot tfsup_j}\cdot A_s^{-1}\cdot z^{-\tau_j}U'_j(s)\end{aligned} \quad (8)$$

According to equation (7), (8), (5) and (6), a simple matrix form can be obtained:

$$
\begin{bmatrix} D_{j+1}(s) \\ D'_{j+1}(s) \\ U_{j+1}(s) \\ U'_{j+1}(s) \end{bmatrix} = \frac{z^{-\tau_j}}{tffup_j \cdot tssup_j - tsfup_j \cdot tfsup_j} \cdot [M_j]_{4\times 4} \cdot \begin{bmatrix} D_j(s) \\ D'_j(s) \\ U_j(s) \\ U'_j(s) \end{bmatrix}, \qquad (9)
$$

here $[M_j]$ is the propagator matrix for Fast P wave and Slow P wave traveling vertically through a porous, permeable fluid-saturated layer *j*. The matrix element of $[M_j]$ can be found from equations (7), (8), (5) and (6), and are summarized in appendix 1. Thus given the asymptotic calculations of the reflection and transmission coefficients for each layer, we could recursively calculate all the downgoing and upgoing waveforms $D_j(s)$, $D_j'(s)$, $U_j(s)$ and $U_j'(s)$ in Laplace transforms from layer 0 to layer k+1, where layer 0 and layer k+1 are two half spaces (Figure 6).

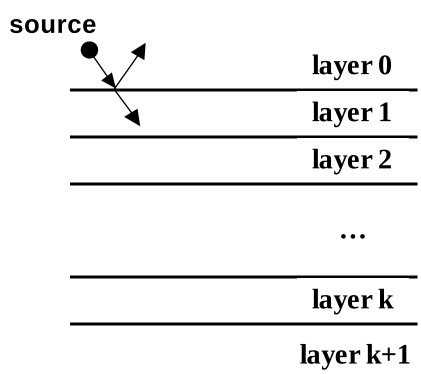

Figure 6. Schematic plot of the propagations of waves through multi-layered media from layer 0 to layer k+1.

From layer 1 to layer k+1, matrix propagation provides the following relation:

$$\begin{bmatrix} D_{k+1}(s) \\ D'_{k+1}(s) \\ U_{k+1}(s) \\ U'_{k+1}(s) \end{bmatrix} = \frac{z^{-\sum_{j=1}^{k}\tau_j}}{\prod_{j=1}^{k}(tffup_j \cdot tssup_j - tsfup_j \cdot tfsup_j)} \cdot M_k ... M_2 M_1 \cdot \begin{bmatrix} D_1(s) \\ D'_1(s) \\ U_1(s) \\ U'_1(s) \end{bmatrix},$$

here, M_1, M_2, …, M_k are the propagator matrix of layer 1, 2,…, k.

If we assume that the source is placed on the surface, i.e., the bottom of layer 0, then there is no time delay from layer 0 to layer 1. Thus, by setting the time delay $z = 1$ in equation (9), we can obtain the waveforms from layer 0 to layer 1 by:

$$\begin{bmatrix} D_1(s) \\ D'_1(s) \\ U_1(s) \\ U'_1(s) \end{bmatrix} = \frac{1}{tffup_0 \cdot tssup_0 - tsfup_0 \cdot tfsup_0} \cdot M_0 \cdot \begin{bmatrix} D_0(s) \\ D'_0(s) \\ U_0(s) \\ U'_0(s) \end{bmatrix},$$

here, M_0 is the propagator matrix of layer 0. Together with the matrix propagation from layer 1 to layer k+1, we could derive a relation between the waveforms of layer 0 and layer k+1 by the following equation:

$$\begin{bmatrix} D_{k+1}(s) \\ D'_{k+1}(s) \\ U_{k+1}(s) \\ U'_{k+1}(s) \end{bmatrix} = \frac{z^{-\sum_{j=1}^{k}\tau_j}}{\prod_{j=0}^{k}(tffup_j \cdot tssup_j - tsfup_j \cdot tfsup_j)} \cdot \prod_{j=0}^{k} M_j \cdot \begin{bmatrix} D_0(s) \\ D'_0(s) \\ U_0(s) \\ U'_0(s) \end{bmatrix} = C \cdot N_k \cdot \begin{bmatrix} D_0(s) \\ D'_0(s) \\ U_0(s) \\ U'_0(s) \end{bmatrix},$$

where,

$$C = \frac{z^{-\sum_{j=1}^{k}\tau_j}}{\prod_{j=0}^{k}(tffup_j \cdot tssup_j - tsfup_j \cdot tfsup_j)} \text{ ; and } N_k = \prod_{j=0}^{k} M_j .$$

Matrix M_j and M_0 have the following forms and the explicit expressions can be found in appendix 1:

$$M_j = \begin{bmatrix} M_j(1,1) & M_j(1,2) & M_j(1,3) & M_j(1,4) \\ M_j(2,1) & M_j(2,2) & M_j(2,3) & M_j(2,4) \\ M_j(3,1) & M_j(3,2) & M_j(3,3) & M_j(3,4) \\ M_j(4,1) & M_j(4,2) & M_j(4,3) & M_j(4,4) \end{bmatrix},$$

and,

$$M_0 = \begin{bmatrix} M_0(1,1) & M_0(1,2) & M_0(1,3) & M_0(1,4) \\ M_0(2,1) & M_0(2,2) & M_0(2,3) & M_0(2,4) \\ M_0(3,1) & M_0(3,2) & M_0(3,3) & M_0(3,4) \\ M_0(4,1) & M_0(4,2) & M_0(4,3) & M_0(4,4) \end{bmatrix}.$$

3.2 Reflectivity and transmitivity series

We already obtained the matrix propagation from layer 0 to layer k+1, thus we have the equation below:

$$\begin{bmatrix} D_{k+1}(s) \\ D'_{k+1}(s) \\ U_{k+1}(s) \\ U'_{k+1}(s) \end{bmatrix} = C \cdot N_k \cdot \begin{bmatrix} D_0(s) \\ D'_0(s) \\ U_0(s) \\ U'_0(s) \end{bmatrix} = C \cdot \begin{bmatrix} N_k(1,1) & N_k(1,2) & N_k(1,3) & N_k(1,4) \\ N_k(2,1) & N_k(2,2) & N_k(2,3) & N_k(2,4) \\ N_k(3,1) & N_k(3,2) & N_k(3,3) & N_k(3,4) \\ N_k(4,1) & N_k(4,2) & N_k(4,3) & N_k(4,4) \end{bmatrix} \cdot \begin{bmatrix} D_0(s) \\ D'_0(s) \\ U_0(s) \\ U'_0(s) \end{bmatrix},$$

where, matrix N_k is obtained by recursive multiplication of propagator matrix M_0 through M_k, i.e.,

$$N_k = M_k ... M_1 \cdot M_0 = \prod_{j=0}^{k} M_j \; .$$

Note that each matrix element of propagator matrix M_j is a polynomial in z (Appendix 1), for example, $M_j(1,3) = (rffup_j \cdot tssup_j - rsfup_j \cdot tfsup_j) \cdot A_f^{-1} \cdot z^{\tau_j - 1}$, which has only ($\tau_j$-1)-th order term in z polynomial. And notice that the highest order non-zero term in matrix M_j has ($2\tau_j$)-th order, they are $M_j(1,2)$, $M_j(2,2)$, $M_j(3,2)$ and $M_j(4,2)$, thus we can define an array with length of ($2\tau_j$+1) to represent the matrix element of M_j, where the first value of the array corresponds to the zero order term in z polynomial, and the ($2\tau_j$+1)-th value of the array corresponds to the ($2\tau_j$)-th order term in z polynomial. The multiplication of each matrix element of M_j is equivalent to convolution of its corresponding arrays. Thus, all the matrix operations can be applied to our matrix propagation process except for replacing the multiplication of matrix elements by a convolution of the corresponding arrays.

The impulse input corresponds to a downgoing Fast P wave with amplitude equals to 1, i.e., $D_0(s) = 1$ and we assume there is no input Slow P wave, thus, $D_0' = 0$. And there are also no upgoing waveforms from the bottom half space (layer k+1), thus, $U_{k+1} = 0$ and $U_{k+1}' = 0$. This gives:

$$\begin{bmatrix} D_{k+1}(s) \\ D'_{k+1}(s) \\ 0 \\ 0 \end{bmatrix} = C \cdot N_k \cdot \begin{bmatrix} D_0(s) \\ D'_0(s) \\ U_0(s) \\ U'_0(s) \end{bmatrix} = C \cdot \begin{bmatrix} N_k(1,1) & N_k(1,2) & N_k(1,3) & N_k(1,4) \\ N_k(2,1) & N_k(2,2) & N_k(2,3) & N_k(2,4) \\ N_k(3,1) & N_k(3,2) & N_k(3,3) & N_k(3,4) \\ N_k(4,1) & N_k(4,2) & N_k(4,3) & N_k(4,4) \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ U_0(s) \\ U'_0(s) \end{bmatrix}.$$

Expand the above equation, we have:

$$\begin{cases} 0 = N_k(3,1) + N_k(3,3)\cdot U_0(s) + N_k(3,4)\cdot U_0^{'}(s) \\ 0 = N_k(4,1) + N_k(4,3)\cdot U_0(s) + N_k(4,4)\cdot U_0^{'}(s) \end{cases}.$$

Solve these two equations for $U_0(s)$ and $U_0'(s)$, we obtain:

$$U_0(s) = \frac{N_k(4,4)\cdot N_k(3,1) - N_k(3,4)\cdot N_k(4,1)}{N_k(3,4)\cdot N_k(4,3) - N_k(3,3)\cdot N_k(4,4)}, \text{ and}$$

$$U_0^{'}(s) = \frac{N_k(3,3)\cdot N_k(4,1) - N_k(3,1)\cdot N_k(4,3)}{N_k(3,4)\cdot N_k(4,3) - N_k(3,3)\cdot N_k(4,4)} .$$

$U_0(s)$ and $U_0'(s)$ as polynomials of z, are the reflectivity series of P wave and Slow wave, respectively.

We also have the equation for $D_{k+1}(s)$ as follows:

$$D_{k+1}(s) = C\cdot[N_k(1,1) + N_k(1,3)\cdot U_0(s) + N_k(1,4)\cdot U_0^{'}(s)],$$

where,

$$\mathrm{C} = \frac{z^{-\sum_{j=0}^{k}\tau_j}}{\prod_{j=0}^{k}(tffup_j\cdot tssup_j - tsfup_j\cdot tfsup_j)}$$

Substitute $U_0(s)$ and $U_0'(s)$ into this equation for $D_{k+1}(s)$, we obtain:

$$\begin{aligned} D_{k+1}(s) = C\cdot\Bigg[& \frac{N_k(1,1)\cdot N_k(3,4)\cdot N_k(4,3) - N_k(1,1)\cdot N_k(3,3)\cdot N_k(4,4)}{N_k(3,4)\cdot N_k(4,3) - N_k(3,3)\cdot N_k(4,4)} + \\ & \frac{N_k(1,3)\cdot N_k(4,4)\cdot N_k(3,1) - N_k(1,3)\cdot N_k(3,4)\cdot N_k(4,1)}{N_k(3,4)\cdot N_k(4,3) - N_k(3,3)\cdot N_k(4,4)} + \\ & \frac{N_k(1,4)\cdot N_k(3,3)\cdot N_k(4,1) - N_k(1,4)\cdot N_k(3,1)\cdot N_k(4,3)}{N_k(3,4)\cdot N_k(4,3) - N_k(3,3)\cdot N_k(4,4)}\Bigg]. \end{aligned}$$

$D_{k+1}(s)$ as a polynomial of z is the transmitivity series of the impulse P wave propagating through a multi-layered media.

3.2 Results of calculations

Figure 7 shows the reflectivity series vs. frequency from seven layers of rock with inhomogeneous fluid saturation (Table 5) and Figure 8 shows the same plot for a homogeneous fluid saturation with slightly different rock properties (Table 6). We can see a remarkable influence of the Slow P wave for the rock with inhomogeneous fluid saturation, while for the rock with homogeneous fluid saturation Slow P wave effect is very small. This is consistent with previous studies (Dutta & Ode, 1979a, b; Carcione et al., 2003).

It should be noted that previous studies recognized the high attenuation in the inhomogeneous fluid-saturated rock is due to the energy flow to Slow P wave, while in our case, the signal from mode conversion to Slow P wave is directly calculated.

Also, we can see in general the lower frequency signals last longer than higher frequencies. This phenomenon is similar to the low frequency shadows that are often observed beneath gas reservoirs. Since gas reservoir would introduce some degree of inhomogeneity in fluid saturation, such as gas bubbles. According to Figure 5, we think Slow wave may be a major cause for the low frequency shadows.

Table 5. Input rock and fluid properties. Rock properties are the same for each layer, while fluid properties are changed alternatively between gas and water. Layer thickness equals to 0.1 ms for P wave.

Grain bulk modulus	Grain density	Dry rock bulk modulus	Dry rock shear modulus	Porosity	Permeab.	Fluid bulk modulus	Fluid density	Fluid visc-osity
K_g [Gpa]	ρ_g [g/cc]	K_{dry} [Gpa]	μ_{dry} [Gpa]	ϕ	κ [darcy]	K_f [Gpa]	ρ_f [g/cc]	η_f [cp]
38	2.65	1.46	1.56	0.3	2	0.025	0.15	0.01
38	2.65	1.46	1.56	0.3	2	2.42	1	1
38	2.65	1.46	1.56	0.3	2	0.025	0.15	0.01
…	...	…	...	…	…	…	...	…
38	2.65	1.46	1.56	0.3	2	0.025	0.15	0.01

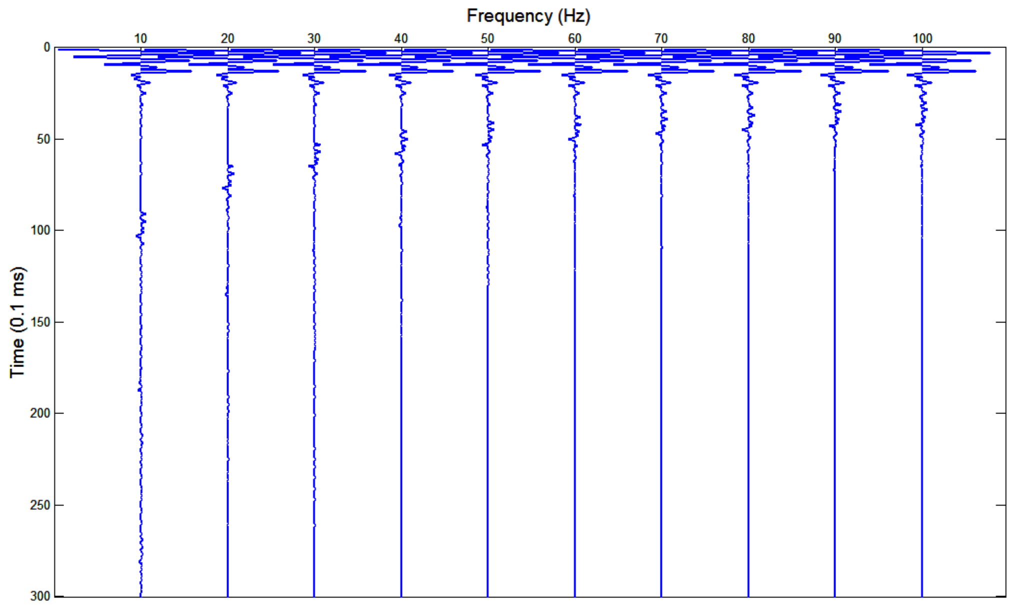

Figure 7. Reflectivity series vs. frequency from seven layers of porous permeable sands with gas and water saturated alternatively (Table 5).

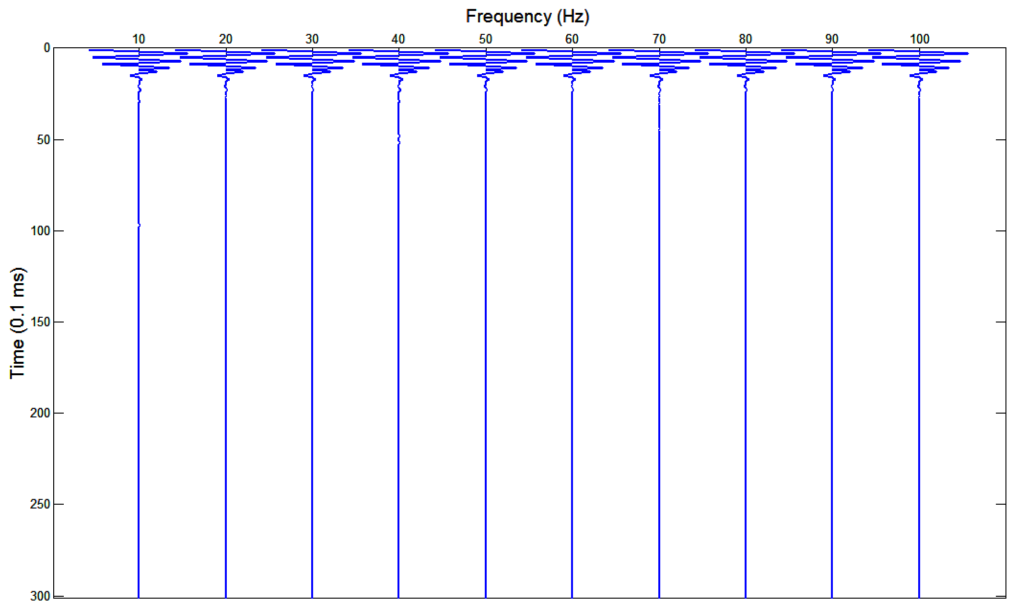

Figure 8. Reflectivity series vs. frequency from seven layers of porous permeable sands with only water saturation (Table 6).

Table 6. Input rock and fluid properties. Fluid properties are the same for each layer, rock properties are changed alternatively. Layer thickness equals to 0.1 ms for P wave.

Grain bulk modulus	Grain density	Dry rock bulk modulus	Dry rock shear modulus	Porosity	Permeab.	Fluid bulk modulus	Fluid density	Fluid visco-sity
K_g [Gpa]	ρ_g [g/cc]	K_{dry} [Gpa]	μ_{dry} [Gpa]	ϕ	κ [darcy]	K_f [Gpa]	ρ_f [g/cc]	η_f [cp]
38	2.65	1.46	1.56	0.3	2	2.42	1	1
35	2.65	1.7	1.855	0.1	0.1	2.42	1	1
38	2.65	1.46	1.56	0.3	2	2.42	1	1
…	...	…	...	…	…	…	...	…
38	2.65	1.46	1.56	0.3	2	2.42	1	1

The number of very thin fluid-saturated permeable layers has also effect on the seismic response. Figure 9 shows reflectivity series from nine layers of alternative gas/water sands, we can see that the seismic signal is enhanced as compared to Figure 7.

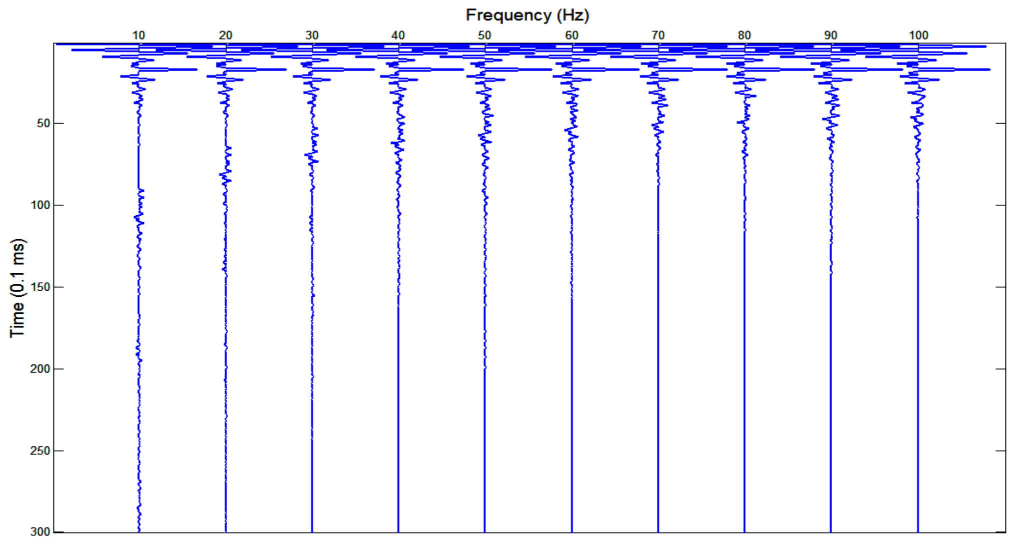

Figure 9. Same with Figure 7, only the number of total layers equals to nine.

In conclusion we would like to mark: 1) low frequency asymptotic description of the Biot's model was applied based on matrix propagator technique; 2) the asymptotic description provides accurate calculations at seismic frequencies; 3) Slow P wave effect strongly depends on frequency and in case of a reservoir with high permeability and inhomogeneous fluid saturation the Slow P wave effect must be taken into account.

CHAPTER 4: Future Directions

4.1 Improvement of propagator matrix method

Although the basic theory and matrix element of the propagator matrix for Fast-Slow P wave conversion have been derived, numerical instability due the limitation of the computer precision will lead to very serious problems. These problems make the application of the Propagator Matrix method only suitable for very thin layer and high permeable rocks. Jocker et al (2004) studied the numerical instability for propagator matrix method applied to Biot's model. In order to solve this numerical instability problem, some modification on the algorithm is needed. Studies of Dunkin (1965), Schmidt and Tango (1986), Levesque and Piche (1992) suggested some modification methods. Thus, to improve the numerical stability of the propagator matrix method applied in this study can be one of the future directions.

4.2 Permeability attributes

Based on the Asymptotic algorithm in Goloshubin and Silin (2005), a relative permeability attribute has been derived and applied on seismic data analysis. In Focus software version 5.4, PERMATR module allows users to convert the seismic amplitude traces into relative permeability traces. Examples of how PERMATR works are shown in Figure 10 and Figure 11. The seismic data is from South Marsh, Gulf of Mexico. There is a water well (SM_238_88) located at INLINE 2170, XLINE 1033. We can see the reservoir zone at about 2.7 second from Figure 10 and Figure 11. The high permeable sand is more apparent from the relative permeability traces. Thus, another possible future project can be on mapping of the high permeability zone from seismic data using PERMATR module, convert the seismic amplitude trace into permeability trace and find out the difference between them.

Future project related to this permeability attribute can also be trying to improve the permeability attributes and make it more reliable and more accurate.

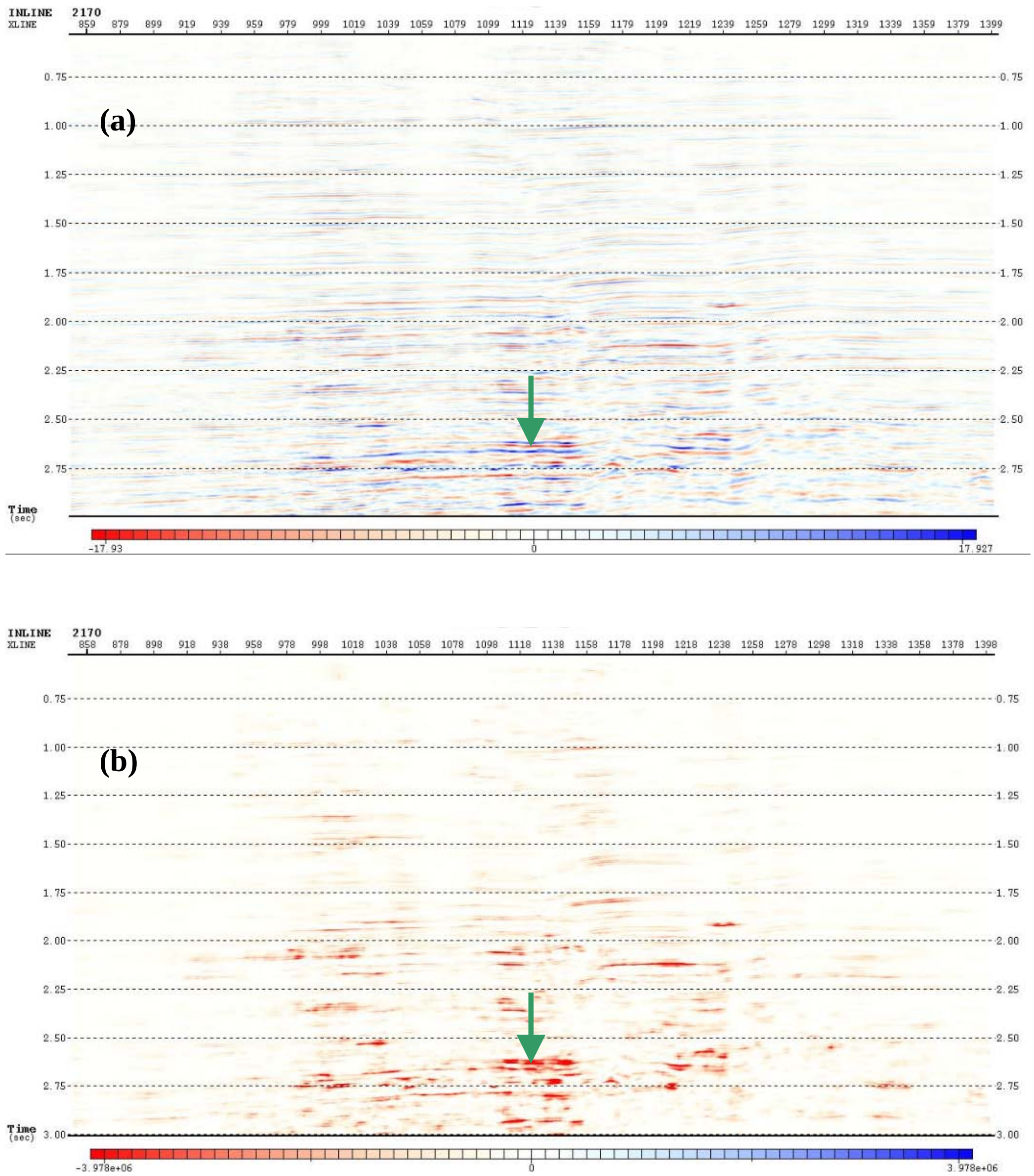

Figure 10. (a) The seismic amplitude trace for INLINE 2170, (b) the relative permeability trace for INLINE 2170. The reservoir zone is indicated by arrow.

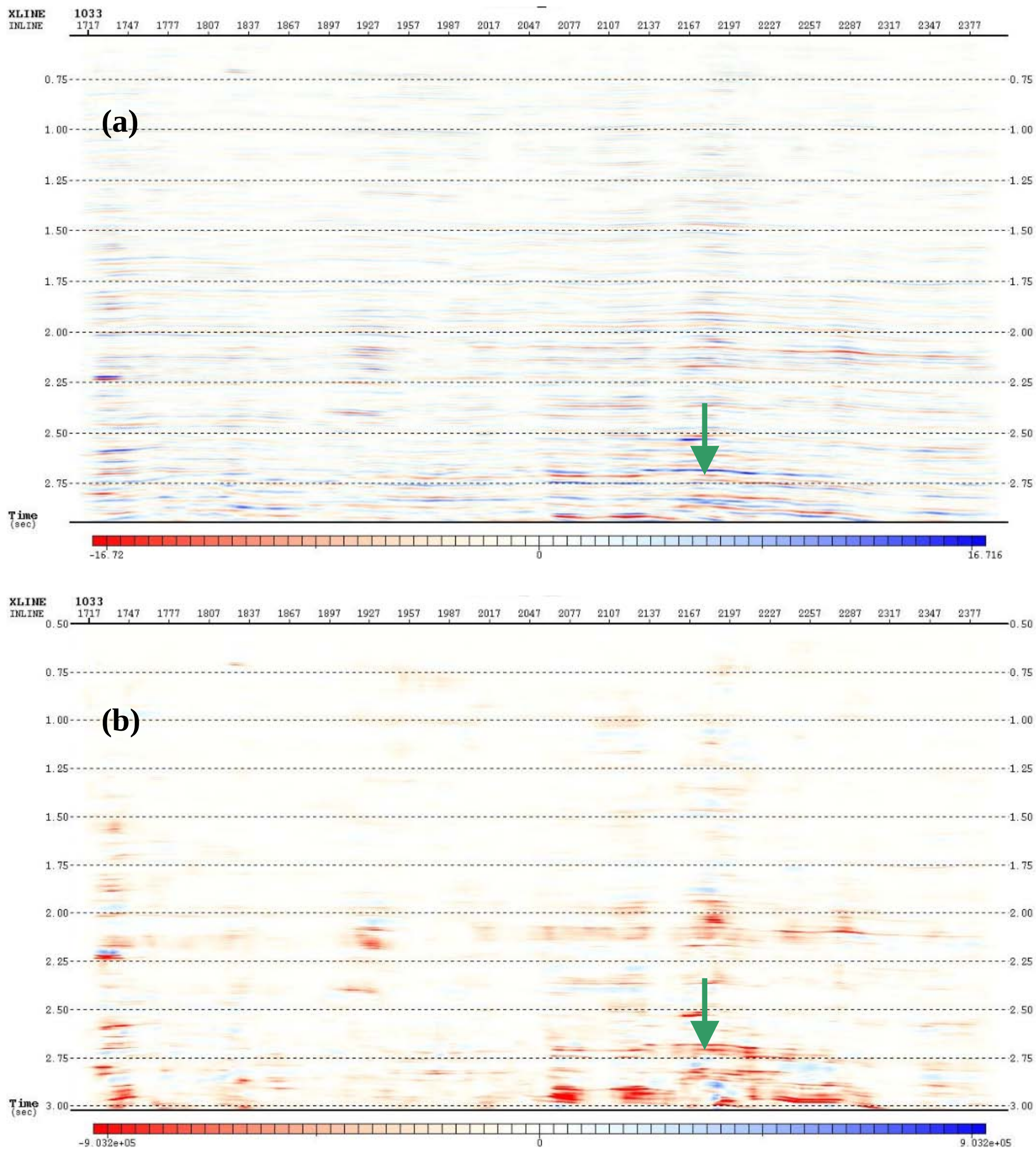

Figure 11. (a) The seismic amplitude trace for XLINE 1033, (b) the relative permeability trace for XLINE 1033. The reservoir zone is indicated by arrow.

References

Biot, M. A., 1956a, Theory of propagation of elastic waves in a fluid-saturated porous solid. I. Low-frequency range: Journal of the Acoustical Society of America, **28**, 168 – 178.

Biot, M. A., 1956b, Theory of propagation of elastic waves in a fluid-saturated porous solid. II. Higher frequency range: Journal of the Acoustical Society of America, **28**, 179 – 191.

Carcione, J. M., Helle, H. B., and Pham, N. H., 2003, White's model for wave propagation in partially saturated rocks: Comparison with poroelastic numerical experiments: Geophysics, **68**, 1389 – 1398.

Castagna, J. P., Sun, S., and Siegfried, R. W., 2003, Instantaneous spectral analysis: Detection of low-frequency shadows associated with hydrocarbons: The Leading Edge, **22**, 120 – 27.

Dunkin, J. W., 1965, Computation of modal solutions in layered, elastic media at high frequencies: Bulletin of Seismological Society of America, **55**, 335 – 358.

Dutta, N. C., and Ode, H., 1983, Seismic reflections from a gas-water contact: Geophysics, **48**, 148 – 162.

Dutta, N. C., and Ode, H., 1979a, Attenuation and dispersion of compressional waves in fluid-filled porous rocks with partial gas saturation (White model)-Part I: Biot theory: Geophysics, **44**, 1777 – 1788.

Dutta, N. C., and Ode, H., 1979b, Attenuation and dispersion of compressional waves in fluid-filled porous rocks with partial gas saturation (White model)-Part II: Results: Geophysics, **44**, 1789 – 1805.

Goloshubin, G., Silin, D., Vingalov, V., and Latfullin, L., 2008, Reservoir permeability from seismic attribute analysis: The Leading Edge, **27**, 376 – 381.

Goloshubin, G., Silin, D., 2005, Using frequency-dependent seismic attributes in imaging of a fractured reservoir zone. 75th Annual SEG Meeting, Houston, TX.

Haskell, N., 1953, The dispersion of surface waves in multilayered media: Bulletin of the Seismological Society of America, **43**, 17 – 34.

Jocker, J., Smeulders, D., Drijkoningen, G., van der Lee, C., and Kalfsbeek, A., 2004, Matrix propagator method for layered porous media: Analytical expressions and stability criteria: Geophysics, **69**, 1071 – 1081.

Levesque, D., and Piche, L., 1992, A robust transfer matrix formulation for the ultrasonic response of multilayered absorbing media: Journal of the Acoustical Society of America, **92**, 452 – 467.

Robinson, E. A., 1967, Multichannel time series analysis with digital computer programs: Holden-Day.

Silin, D., and Goloshubin, G., 2008, Seismic wave reflection from a permeable layer: low-frequency asymptotic analysis: Proceedings of IMECE.

Schmidt, H., and Tango, G., 1986, Efficient global matrix approach to the computation of synthetic seismograms: Geophys. J. R. astr. Soc., **84**, 331 – 359.

Thomson, W., 1950, Transmission of elastic waves through a stratified solid medium: Journal of Applied Physics, **21**, 89 – 93.

Appendix 1: Matrix Elements of M_j and M_0

$$M_j(1,1) = [tff_j \cdot (tffup_j \cdot tssup_j - tsfup_j \cdot tfsup_j) + rffup_j \cdot (-tssup_j \cdot rff_j + tsfup_j \cdot rfs_j) + rsfup_j \cdot (-rfs_j \cdot tffup_j + rff_j \cdot tfsup_j)] \cdot A_f \cdot z^{\tau_j + 1}$$

$$M_j(1,2) = [tsf_j \cdot (tffup_j \cdot tssup_j - tsfup_j \cdot tfsup_j) + rffup_j \cdot (-tssup_j \cdot rsf_j + tsfup_j \cdot rss_j) + rsfup_j \cdot (-rss_j \cdot tffup_j + rsf_j \cdot tfsup_j)] \cdot A_s \cdot z^{2\tau_j}$$

$$M_j(1,3) = (rffup_j \cdot tssup_j - rsfup_j \cdot tfsup_j) \cdot A_f^{-1} \cdot z^{\tau_j - 1}$$

$$M_j(1,4) = (-rffup_j \cdot tsfup_j + rsfup_j \cdot tffup_j) \cdot A_s^{-1} \cdot 1$$

$$M_j(2,1) = [tfs_j \cdot (tffup_j \cdot tssup_j - tsfup_j \cdot tfsup_j) + rfsup_j \cdot (-tssup_j \cdot rff_j + tsfup_j \cdot rfs_j) + rssup_j \cdot (-rfs_j \cdot tffup_j + rff_j \cdot tfsup_j)] \cdot A_f \cdot z^{\tau_j + 1}$$

$$M_j(2,2) = [tss_j \cdot (tffup_j \cdot tssup_j - tsfup_j \cdot tfsup_j) + rfsup_j \cdot (-tssup_j \cdot rsf_j + tsfup_j \cdot rss_j) + rssup_j \cdot (-rss_j \cdot tffup_j + rsf_j \cdot tfsup_j)] \cdot A_s \cdot z^{2\tau_j}$$

$$M_j(2,3) = (rfsup_j \cdot tssup_j - rssup_j \cdot tfsup_j) \cdot A_f^{-1} \cdot z^{\tau_j - 1}$$

$$M_j(2,4) = (-rfsup_j \cdot tsfup_j + rssup_j \cdot tffup_j) \cdot A_s^{-1} \cdot 1$$

$$M_j(3,1) = (-tssup_j \cdot rff_j + tsfup_j \cdot rfs_j) \cdot A_f \cdot z^{\tau_j + 1}$$

$$M_j(3,2) = (-tssup_j \cdot rsf_j + tsfup_j \cdot rss_j) \cdot A_s \cdot z^{2\tau_j}$$

$$M_j(3,3) = tssup_j \cdot A_f^{-1} \cdot z^{\tau_j - 1}$$

$$M_j(3,4) = -tsfup_j \cdot A_s^{-1} \cdot 1$$

$$M_j(4,1) = (-rfs_j \cdot tffup_j + rff_j \cdot tfsup_j) \cdot A_f \cdot z^{\tau_j + 1}$$

$$M_j(4,2) = (-rss_j \cdot tffup_j + rsf_j \cdot tfsup_j) \cdot A_s \cdot z^{2\tau_j}$$

$$M_j(4,3) = -tfsup_j \cdot A_f^{-1} \cdot z^{\tau_j - 1}$$

$$M_j(4,4) = tffup_j \cdot A_s^{-1} \cdot 1$$

$$M_0(1,1) = tff_0 \cdot (tffup_0 \cdot tssup_0 - tsfup_0 \cdot tfsup_0) + rffup_0 \cdot (-tssup_0 \cdot rff_0 + tsfup_0 \cdot rfs_0) + rsfup_0 \cdot (-rfs_0 \cdot tffup_0 + rff_0 \cdot tfsup_0)$$

$$M_0(1,2) = tsf_0 \cdot (tffup_0 \cdot tssup_0 - tsfup_0 \cdot tfsup_0) + rffup_0 \cdot (-tssup_0 \cdot rsf_0 + tsfup_0 \cdot rss_0) + rsfup_0 \cdot (-rss_0 \cdot tffup_0 + rsf_0 \cdot tfsup_0)$$

$$M_0(1,3) = rffup_0 \cdot tssup_0 - rsfup_0 \cdot tfsup_0$$

$$M_0(1,4) = -rffup_0 \cdot tsfup_0 + rsfup_0 \cdot tffup_0$$

$$M_0(2,1) = tfs_0 \cdot (tffup_0 \cdot tssup_0 - tsfup_0 \cdot tfsup_0) + rfsup_0 \cdot (-tssup_0 \cdot rff_0 + tsfup_0 \cdot rfs_0) + rssup_0 \cdot (-rfs_0 \cdot tffup_0 + rff_0 \cdot tfsup_0)$$

$$M_0(2,2) = tss_0 \cdot (tffup_0 \cdot tssup_0 - tsfup_0 \cdot tfsup_0) + rfsup_0 \cdot (-tssup_0 \cdot rsf_0 + tsfup_0 \cdot rss_0) + rssup_0 \cdot (-rss_0 \cdot tffup_0 + rsf_0 \cdot tfsup_0)$$

$$M_0(2,3) = rfsup_0 \cdot tssup_0 - rssup_0 \cdot tfsup_0$$

$$M_0(2,4) = -rfsup_0 \cdot tsfup_0 + rssup_0 \cdot tffup_0$$

$$M_0(3,1) = -tssup_0 \cdot rff_0 + tsfup_0 \cdot rfs_0$$

$$M_0(3,2) = -tssup_0 \cdot rsf_0 + tsfup_0 \cdot rss_0$$

$$M_0(3,3) = tssup_0$$

$$M_0(3,4) = -tsfup_0$$

$$M_0(4,1) = -rfs_0 \cdot tffup_0 + rff_0 \cdot tfsup_0$$

$$M_0(4,2) = -rss_0 \cdot tffup_0 + rsf_0 \cdot tfsup_0$$

$$M_0(4,3) = -tfsup_0$$

$$M_0(4,4) = tffup_0$$

Appendix 2: Fortran Code

A2.1 Example input file

```
1
3     22     0.0001 3
38    2.65   1.46   1.56   0.3   2   0.025   0.15   0.01
38    2.65   1.46   1.56   0.3   2   2.42    1      1
38    2.65   1.46   1.56   0.3   2   0.025   0.15   0.01
```

A2.2 Output file from the input above

```
Type 0, to see self-document file

 nlayer    freq[Hz]    dt[sec]    multiples
   3        100.00      0.000100      3

  kg      rhog    kdry    udry    phi     kappa   kf      rhof    nf
  [Gpa]   [g/cc]  [Gpa]   [Gpa]           [darcy] [Gpa]   [g/cc]  [cp]
  35.000  2.650   1.700   1.855   0.300   1.000   0.022   0.100   0.015
  35.000  2.650   1.700   1.855   0.300   1.000   2.400   1.000   1.000

 Reflectivity Slow wave  192
          -0.2657334804534913
           0.0000000000000000
           0.2313374904532081
           0.0000000000000000
           0.0134042316281641
           0.0000000000000000
           ……
Transmitivity Slow wave  267
           0.9610546140364452
           0.0000000000000000
           0.0556857370096958
           0.0000000000000000
           0.0032265609685689
           0.0000000000000000
           ……
```

A2.3 Fortran code (http://www.geosc.uh.edu/~yliu/slowave.for)

```
C---------------------------------------------------------------------------
C      slowave - calculates the 1-D impulse reflectivity and
C                transmitivity series of the Biot's Fast P wave and Slow
C                P wave propagations through the fluid-saturated porous
C                permeable layered media.
C---------------------------------------------------------------------------
C      Input:
C       nlayer        number of total layers
C       freq          frequency in [Hz]
C       dt            sample rate or sample time [sec]
C       mtps          number of multiples (mtps >= 2)
C
C       kg            grain bulk modulus    [Gpa]
C       rhog          grain density         [g/cc]
C       kdry          dry rock bulk modulus         [Gpa]
C       udry          dry rock shear modulus        [Gpa]
C       phi           porosity
C       kappa         permeability          [darcy]
C       kf            fluid bulk modulus    [Gpa]
C       rhof          fluid density         [g/cc]
C       nf            fluid viscosity       [cp]
C
C      Output:
C       a             reflectivity series
C       b             transmitivity series
C       nsr           number of samples in reflectivity series
C       nst           number of samples in transmitivity series
C---------------------------------------------------------------------------
C      Input File Format:
C---------------------------------------------------------------------------
C      1
C      nlayer, freq, dt, mtps
C      kg, rhog, kdry, udry, phi, kappa, kf, rhof, nf  [ Layer 0 ]
C      kg, rhog, kdry, udry, phi, kappa, kf, rhof, nf  [ Layer 1 ]
C                    ......
C      kg, rhog, kdry, udry, phi, kappa, kf, rhof, nf  [ Layer k ]
C---------------------------------------------------------------------------
C      Input File Example:
C---------------------------------------------------------------------------
C       1
C       3       22      0.0001 3
```

```
C        38      2.65     1.46     1.56     0.3    2    0.025    0.15    0.01
C        38      2.65     1.46     1.56     0.3    2    2.42     1         1
C        38      2.65     1.46     1.56     0.3    2    0.025    0.15    0.01
C-----------------------------------------------------------------------------
C       Note:   Layer 0 and Layer k are two half spaces.
C               nlayer = k + 1
C-----------------------------------------------------------------------------
C       References: Silin D., and Goloshubin G., 2008, Seismic
C                     wave reflection from a permeable layer:
C                     low-frequency asymptotic analysis: Proceedings
C                     of IMECE, Boston, USA.
C                   Robinson, E. A., 1967, Multichannel time series
C                     analysis with digital computer programs:
C                     Holden-Day, San Francisco.
C-----------------------------------------------------------------------------
C       Author:         Yangjun (Kevin) Liu  University of Houston2009
C-----------------------------------------------------------------------------
      program slowave
        double precision a(10000), b(10000)
        integer flag

        write(*,*) 'Type 0, to see self-document file'
        read (*,*) flag

        if(flag.eq.0) goto 10
        if(flag.eq.1) goto 20

10      write(*,*) '\n This program calculates reflectivity and'
        write(*,*) 'transmitivity series of P wave and Biot Slow'
        write(*,*) 'wave propagate through fluid-saturated porous'
        write(*,*) 'permeable layered media.'
        write(*,*) '\n An input file is needed.'
        write(*,*) '\n Input file has format below:'
        write(*,*) '1'
        write(*,*) 'nlayer, freq, dt, multiples'
        write(*,*) 'kg, rhog, kdry, udry, phi, kappa, kf, rhof, nf
     + [ Layer 0 ]'
        write(*,*) 'kg, rhog, kdry, udry, phi, kappa, kf, rhof, nf
     + [ Layer 1 ]'
        write(*,*) '                  ......'
        write(*,*) 'kg, rhog, kdry, udry, phi, kappa, kf, rhof, nf
     + [ Layer k ]'
        write(*,*) '\n Example of an input file shows below:'
```

```
      write(*,*) '1'
      write(*,*) '3    22       0.0001 3'
      write(*,*) '38  2.65     1.46     1.56     0.3    2    0.025
     +   0.15    0.01'
      write(*,*) '38  2.65     1.46     1.56     0.3    2    2.42
     +    1       1'
      write(*,*) '38  2.65     1.46     1.56     0.3    2    0.025
     +   0.15    0.01\n'
      write(*,*) 'This self-document also generates an input file:
     + input.dat\n'
      write(*,*) 'To get output, type: a <input.dat >output.dat\n'
      write(*,*) 'Check the output in the new file: output.dat\n'

      open(7,file='input.dat',status='unknown')
      write(7,*) '1'
      write(7,*) '3    22       0.0001 3'
      write(7,*) '38  2.65     1.46     1.56     0.3    2    0.025
     +   0.15    0.01'
      write(7,*) '38  2.65     1.46     1.56     0.3    2    2.42
     +    1       1'
      write(7,*) '38  2.65     1.46     1.56     0.3    2    0.025
     +   0.15    0.01\n'
      pause
      goto 100

20    read(*,*) nlayer, freq, dt, mtps

      write(*,*) '\n nlayer      freq[Hz]      dt[sec]      multiples'
      write(*,30) nlayer, freq, dt, mtps
30    format(i4,5x,f10.2,8x,f8.6,6x,i4)

      lc=nlayer-1               !/* number of subsurface interfaces */

      call refsl(mtps,lc,freq,dt,a,b,nsr,nst)

      write(*,*) '\n Reflectivity Slow wave ', nsr
      call outdat(nsr,a)
      write(*,*) '\n Transmitivity Slow wave ', nst
      call outdat(nst,b)

100   stop

      end
```

```
C-------------------------------------------------------------------------
C       refsl - calculate reflectivity and transmitivity series taken
C                    into account Biot's slow wave.
C-------------------------------------------------------------------------
C       Input:
C        mtps   number of multiples (mtps >= 2)
C        lc       number of subsurface interfaces
C        freq    freqeuncy [Hz]
C        dt       sample rate or sample time [sec]
C
C       Output:
C        a[]      reflectivity series
C        b[]      transmitivity series
C        nsr      number of samples in reflectivity series
C        nst      number of samples in transmitivity series
C-------------------------------------------------------------------------
    subroutine refsl(mtps,lc,freq,dt,a,b,nsr,nst)
        double precision c(4,4),c2,cc,a(10000),b(10000)
        double precision p1(10000),p2(10000),q1(10000),q2(10000),p(10000)
        double precision q(10000),ps1(10000),ps2(10000)
        double precision pp1(10000),pp2(10000),pps1(10000),pps2(10000)
        double precision pp(10000),qq1(10000),qq2(10000)
        double precision kg(2),rhog(2),kdry(2),udry(2),phi(2),kappa(2)
        double precision kf(2),rhof(2),nf(2)
        double precision n11(10000),n12(10000),n13(10000),n14(10000)
        double precision n21(10000),n22(10000),n23(10000),n24(10000)
        double precision n31(10000),n32(10000),n33(10000),n34(10000)
        double precision n41(10000),n42(10000),n43(10000),n44(10000)
        integer tau       !/* tau = vfast / vslow */

        cc=1              !/* record the coefficient outside the matrix */
        lse=0             !/* record the number of zeros in reflectivity */
        lsee=0            !/* record the number of zeros in transmitivity */
        dt_e=0            !/* time thickness for top layer (half space) */

        read(*,*) kg(1),rhog(1),kdry(1),udry(1),phi(1),kappa(1),kf(1),
   +     rhof(1),nf(1)
        read(*,*) kg(2),rhog(2),kdry(2),udry(2),phi(2),kappa(2),kf(2),
   +     rhof(2),nf(2)
        write(*,*) '\n  kg       rhog    kdry    udry    phi     kappa   kf
   +     rhof    nf'
        write(*,*) ' [Gpa]   [g/cc]   [Gpa]   [Gpa]          [darcy]  [Gpa]
```

```
     +  [g/cc]  [cp]'
        write(*,10) kg(1),rhog(1),kdry(1),udry(1),phi(1),kappa(1),kf(1),
     +   rhof(1),nf(1)
        write(*,10) kg(2),rhog(2),kdry(2),udry(2),phi(2),kappa(2),kf(2),
     +   rhof(2),nf(2)
10      format(9f8.3)
        call onelayer(kg,rhog,kdry,udry,phi,kappa,kf,rhof,nf,freq,dt_e,c,c2,
     +   tau)
        cc=cc*c2

        ln=1             !/* ln is the length of polynomials in matrix N_k */
                         !/* N_0 = M_0, so ln = 1 for N_0; N_1 = M_1 * N_0 */
        n11(1)=c(1,1)  !\
        n12(1)=c(1,2)  !|
        n13(1)=c(1,3)  !|
        n14(1)=c(1,4)  !|
        n21(1)=c(2,1)  !|
        n22(1)=c(2,2)  !|
        n23(1)=c(2,3)  !|
        n24(1)=c(2,4)  ! > /* form matrix N_0, which equals to M_0 */
        n31(1)=c(3,1)  !|
        n32(1)=c(3,2)  !|
        n33(1)=c(3,3)  !|
        n34(1)=c(3,4)  !|
        n41(1)=c(4,1)  !|
        n42(1)=c(4,2)  !|
        n43(1)=c(4,3)  !|
        n44(1)=c(4,4)  !/

        do k=2, lc       !/* recursively calculate matrix N_k;
                         !   N_k = M_k * N_(k-1) */

         kg(1)=kg(2)
         rhog(1)=rhog(2)
         kdry(1)=kdry(2)
         udry(1)=udry(2)
         phi(1)=phi(2)
         kappa(1)=kappa(2)
         kf(1)=kf(2)
         rhof(1)=rhof(2)
         nf(1)=nf(2)

         read(*,*) kg(2),rhog(2),kdry(2),udry(2),phi(2),kappa(2),kf(2),
```

```
+     rhof(2),nf(2)
       write(*,10) kg(2),rhog(2),kdry(2),udry(2),phi(2),kappa(2),kf(2),
+     rhof(2),nf(2)

       call onelayer(kg,rhog,kdry,udry,phi,kappa,kf,rhof,nf,freq,dt,c,
+     c2,tau)

       cc=cc*c2

       lse=lse+tau-1           !/* the zero coefficients in reflectivity
                               !     polynomials increase with tau-1 */

       lsee=lsee+2*tau         !/* the zero coefficients in transmitivity
                               !     polynomials increase with 2*tau */

       lm=mtps*tau+1                   !/* record the length of polynomials in
                               !     propagator matrix M_k */

       call mprod(c,tau,lm,ln,n11,n12,n13,n14,n21,n22,n23,n24,n31,n32,
+     n33,n34,n41,n42,n43,n44)

      end do

      call fold(ln,n44,ln,n31,lp,p1)
      call fold(ln,n41,ln,n34,lp,p2)
      call fold(ln,n33,ln,n41,lp,ps1)
      call fold(ln,n31,ln,n43,lp,ps2)

      call fold(ln,n13,lp,p1,lpp,pp1)
      call fold(ln,n13,lp,p2,lpp,pp2)
      call fold(ln,n14,lp,ps1,lpp,pps1)
      call fold(ln,n14,lp,ps2,lpp,pps2)

      call fold(ln,n43,ln,n34,lq,q1)
      call fold(ln,n44,ln,n33,lq,q2)
      call fold(ln,n11,lq,q1,lqq,qq1)
      call fold(ln,n11,lq,q2,lqq,qq2)

      do i=1,lpp
       pp(i)=qq1(i)-qq2(i)+pp1(i)-pp2(i)+pps1(i)-pps2(i)
      end do

      call pcut(lpp,pp,lsee)             !/* cut the zero coefficients
```

```
                                        !     in polynomial pp[] */

        do i=1,lq
          q(i)=q1(i)-q2(i)
        end do

        call pcut(lq,q,lse)             !/* cut the zero coefficients
                                        !     in polynomial q[] */

        do i=1,lp
          p(i)=p1(i)-p2(i)
        end do

        call pcut(lp,p,lse)             !/* cut the zero coefficients
                                        !     in polynomial p[] */

C       /* reflectivity series, receiver in the top layer (half space) */

        call polydv(lq,q,lp,p,lp,a)
        nsr=lp

C       /* transmitivity series, receiver in the bottom layer (half space) */

        call polydv(lq,q,lpp,pp,lpp,b)

        do i=1,lpp
          b(i)=b(i)/cc
        end do

        nst=lpp

      return
      end

C---------------------------------------------------------------------------
C       mprod - carry out matrix multiplication: N_k = M_k * N_(k-1)
C---------------------------------------------------------------------------
C       Input:  matrix N_(k-1)
C        c[]    array of coefficients in propagator matrix M
C        tau    slow wave time delay factor, tau = vfast / vslow
C        lm     length of polynomials in propagator matrix M_k
C        ln     length of polynomials in matrix N_(k-1)
C    n11 - n44  polynomials in matrix N_(k-1)
```

```
C
C        Output:          matrix N_k
C         ln       length of polynomials in matrix N_k
C    n11 - n44 polynomials of the matrix elements in matrix N_k
C-----------------------------------------------------------------------
C        Note:   multiplication of polynomials is essentially a
C                   convolution of their coefficients
C-----------------------------------------------------------------------
      subroutine mprod(c,tau,lm,ln,n11,n12,n13,n14,n21,n22,n23,
     +  n24,n31,n32,n33,n34,n41,n42,n43,n44)

        integer tau
        double precision c(4,4)
        double precision n11(10000),n12(10000),n13(10000),n14(10000)
        double precision n21(10000),n22(10000),n23(10000),n24(10000)
        double precision n31(10000),n32(10000),n33(10000),n34(10000)
        double precision n41(10000),n42(10000),n43(10000),n44(10000)
        double precision m11(10000),m12(10000),m13(10000),m14(10000)
        double precision m21(10000),m22(10000),m23(10000),m24(10000)
        double precision m31(10000),m32(10000),m33(10000),m34(10000)
        double precision m41(10000),m42(10000),m43(10000),m44(10000)
        double precision n11a(10000),n12a(10000),n13a(10000),n14a(10000)
        double precision n21a(10000),n22a(10000),n23a(10000),n24a(10000)
        double precision n31a(10000),n32a(10000),n33a(10000),n34a(10000)
        double precision n41a(10000),n42a(10000),n43a(10000),n44a(10000)
        double precision n11b(10000),n12b(10000),n13b(10000),n14b(10000)
        double precision n21b(10000),n22b(10000),n23b(10000),n24b(10000)
        double precision n31b(10000),n32b(10000),n33b(10000),n34b(10000)
        double precision n41b(10000),n42b(10000),n43b(10000),n44b(10000)
        double precision n11c(10000),n12c(10000),n13c(10000),n14c(10000)
        double precision n21c(10000),n22c(10000),n23c(10000),n24c(10000)
        double precision n31c(10000),n32c(10000),n33c(10000),n34c(10000)
        double precision n41c(10000),n42c(10000),n43c(10000),n44c(10000)
        double precision n11d(10000),n12d(10000),n13d(10000),n14d(10000)
        double precision n21d(10000),n22d(10000),n23d(10000),n24d(10000)
        double precision n31d(10000),n32d(10000),n33d(10000),n34d(10000)
        double precision n41d(10000),n42d(10000),n43d(10000),n44d(10000)

        call zero(lm,m11)       !\
        m11(tau+2)=c(1,1)       !|
        call zero(lm,m12)       !|
        m12(2*tau+1)=c(1,2)     !|
        call zero(lm,m13)       !|
```

```
m13(tau)=c(1,3)          !|
call zero(lm,m14)        !|
m14(1)=c(1,4)            !|
call zero(lm,m21)        !|
m21(tau+2)=c(2,1)        !|
call zero(lm,m22)        !|
m22(2*tau+1)=c(2,2)  !|
call zero(lm,m23)        !|
m23(tau)=c(2,3)          !|
call zero(lm,m24)        !|
m24(1)=c(2,4)            ! > /* form propagator matrix M_k */
call zero(lm,m31)        !|
m31(tau+2)=c(3,1)        !|
call zero(lm,m32)        !|
m32(2*tau+1)=c(3,2)  !|
call zero(lm,m33)        !|
m33(tau)=c(3,3)          !|
call zero(lm,m34)        !|
m34(1)=c(3,4)            !|
call zero(lm,m41)        !|
m41(tau+2)=c(4,1)        !|
call zero(lm,m42)        !|
m42(2*tau+1)=c(4,2)  !|
call zero(lm,m43)        !|
m43(tau)=c(4,3)          !|
call zero(lm,m44)        !|
m44(1)=c(4,4)            !/

call fold(lm,m11,ln,n11,j,n11a)
call fold(lm,m12,ln,n21,j,n11b)
call fold(lm,m13,ln,n31,j,n11c)
call fold(lm,m14,ln,n41,j,n11d)
call fold(lm,m11,ln,n12,j,n12a)
call fold(lm,m12,ln,n22,j,n12b)
call fold(lm,m13,ln,n32,j,n12c)
call fold(lm,m14,ln,n42,j,n12d)
call fold(lm,m11,ln,n13,j,n13a)
call fold(lm,m12,ln,n23,j,n13b)
call fold(lm,m13,ln,n33,j,n13c)
call fold(lm,m14,ln,n43,j,n13d)
call fold(lm,m11,ln,n14,j,n14a)
call fold(lm,m12,ln,n24,j,n14b)
call fold(lm,m13,ln,n34,j,n14c)
```

```
call fold(lm,m14,ln,n44,j,n14d)

call fold(lm,m21,ln,n11,j,n21a)
call fold(lm,m22,ln,n21,j,n21b)
call fold(lm,m23,ln,n31,j,n21c)
call fold(lm,m24,ln,n41,j,n21d)
call fold(lm,m21,ln,n12,j,n22a)
call fold(lm,m22,ln,n22,j,n22b)
call fold(lm,m23,ln,n32,j,n22c)
call fold(lm,m24,ln,n42,j,n22d)
call fold(lm,m21,ln,n13,j,n23a)
call fold(lm,m22,ln,n23,j,n23b)
call fold(lm,m23,ln,n33,j,n23c)
call fold(lm,m24,ln,n43,j,n23d)
call fold(lm,m21,ln,n14,j,n24a)
call fold(lm,m22,ln,n24,j,n24b)
call fold(lm,m23,ln,n34,j,n24c)
call fold(lm,m24,ln,n44,j,n24d)

call fold(lm,m31,ln,n11,j,n31a)
call fold(lm,m32,ln,n21,j,n31b)
call fold(lm,m33,ln,n31,j,n31c)
call fold(lm,m34,ln,n41,j,n31d)
call fold(lm,m31,ln,n12,j,n32a)
call fold(lm,m32,ln,n22,j,n32b)
call fold(lm,m33,ln,n32,j,n32c)
call fold(lm,m34,ln,n42,j,n32d)
call fold(lm,m31,ln,n13,j,n33a)
call fold(lm,m32,ln,n23,j,n33b)
call fold(lm,m33,ln,n33,j,n33c)
call fold(lm,m34,ln,n43,j,n33d)
call fold(lm,m31,ln,n14,j,n34a)
call fold(lm,m32,ln,n24,j,n34b)
call fold(lm,m33,ln,n34,j,n34c)
call fold(lm,m34,ln,n44,j,n34d)

call fold(lm,m41,ln,n11,j,n41a)
call fold(lm,m42,ln,n21,j,n41b)
call fold(lm,m43,ln,n31,j,n41c)
call fold(lm,m44,ln,n41,j,n41d)
call fold(lm,m41,ln,n12,j,n42a)
call fold(lm,m42,ln,n22,j,n42b)
call fold(lm,m43,ln,n32,j,n42c)
```

```
call fold(lm,m44,ln,n42,j,n42d)
call fold(lm,m41,ln,n13,j,n43a)
call fold(lm,m42,ln,n23,j,n43b)
call fold(lm,m43,ln,n33,j,n43c)
call fold(lm,m44,ln,n43,j,n43d)
call fold(lm,m41,ln,n14,j,n44a)
call fold(lm,m42,ln,n24,j,n44b)
call fold(lm,m43,ln,n34,j,n44c)
call fold(lm,m44,ln,n44,j,n44d)

do k=1,j
  n11(k)=n11a(k)+n11b(k)+n11c(k)+n11d(k)
end do
do k=1,j
  n12(k)=n12a(k)+n12b(k)+n12c(k)+n12d(k)
end do
do k=1,j
  n13(k)=n13a(k)+n13b(k)+n13c(k)+n13d(k)
end do
do k=1,j
  n14(k)=n14a(k)+n14b(k)+n14c(k)+n14d(k)
end do
do k=1,j
  n21(k)=n21a(k)+n21b(k)+n21c(k)+n21d(k)
end do
do k=1,j
  n22(k)=n22a(k)+n22b(k)+n22c(k)+n22d(k)
end do
do k=1,j
  n23(k)=n23a(k)+n23b(k)+n23c(k)+n23d(k)
end do
do k=1,j
  n24(k)=n24a(k)+n24b(k)+n24c(k)+n24d(k)
end do
do k=1,j
  n31(k)=n31a(k)+n31b(k)+n31c(k)+n31d(k)
end do
do k=1,j
  n32(k)=n32a(k)+n32b(k)+n32c(k)+n32d(k)
end do
do k=1,j
  n33(k)=n33a(k)+n33b(k)+n33c(k)+n33d(k)
end do
```

```
          do k=1,j
            n34(k)=n34a(k)+n34b(k)+n34c(k)+n34d(k)
          end do
          do k=1,j
            n41(k)=n41a(k)+n41b(k)+n41c(k)+n41d(k)
          end do
          do k=1,j
            n42(k)=n42a(k)+n42b(k)+n42c(k)+n42d(k)
          end do
          do k=1,j
            n43(k)=n43a(k)+n43b(k)+n43c(k)+n43d(k)
          end do
          do k=1,j
            n44(k)=n44a(k)+n44b(k)+n44c(k)+n44d(k)
          end do

          ln=j     !/* update the length of polynomials in matrix N_k */

      return
      end

C-----------------------------------------------------------------------------
C         outdat - print out the values of array x[lx]
C-----------------------------------------------------------------------------
      subroutine outdat(lx,x)
          double precision x(lx)
          write(*,5) (x(i),i=1,lx)
5         format(f30.16)
      return
      end

C-----------------------------------------------------------------------------
C         pcut - cut the beginning k values of array P[lp]
C-----------------------------------------------------------------------------
      subroutine pcut(lp,p,k)
          double precision p(lp)
          lp=lp-k
          do i=1,lp
            p(i)=p(i+k)
          end do
      return
      end
```

```
C---------------------------------------------------------------------------
C       polydv - polynomials division (Robinson, 1967)
C---------------------------------------------------------------------------
C       Input:
C        n       length of dvs[]
C        dvs[]   array of polynomial coefficients
C        m       length of dvd[]
C        dvd[]   array of polynomial coefficients
C
C       Output:
C        l       length of q[]
C        q[]     array of polynomial coefficients: q[] = dvd[] / dvs[]
C---------------------------------------------------------------------------
C       Note:   q[1] must not equal to 0
C---------------------------------------------------------------------------
      subroutine polydv(n,dvs,m,dvd,l,q)
        double precision dvs(n),dvd(m),q(l)
        call zero (l,q)
        call move(min(m,l),dvd,q)
        do i=1,l
          q(i)=q(i)/dvs(1)
          if(i.eq.l) return
          k=i
          isub=min(n-1,l-i)
          do j=1,isub
            k=k+1
            q(k)=q(k)-q(i)*dvs(j+1)
          end do
        end do
      return
      end

C---------------------------------------------------------------------------
C       fold - polynomial multiplication (Robinson, 1967)
C---------------------------------------------------------------------------
C       Input:
C        la      length of a[]
C        a[]     array of polynomial coefficients
C        lb      length of b[]
C        b[]     array of polynomial coefficients
C
C       Output:
C        lc      length of c[]
```

```
C          c[]      array of polynomial coefficients:
C---------------------------------------------------------------------------
C        Note:   c[] is essentially the convolution of a[] and b[]
C---------------------------------------------------------------------------
      subroutine fold(la,a,lb,b,lc,c)
        double precision a(la),b(lb),c(lc)
        lc=la+lb-1
        call zero(lc,c)
        do i=1,la
          do j=1,lb
            k=i+j-1
            c(k)=c(k)+a(i)*b(j)
          end do
        end do
      return
      end

C---------------------------------------------------------------------------
C        move - copy the values of array x to array y (Robinson, 1967)
C---------------------------------------------------------------------------
C        Input:
C          lx       length of x[] and y[]
C          x[]      array of polynomial coefficients
C
C        Output:
C          y[]      array of polynomial coefficients
C---------------------------------------------------------------------------
      subroutine move(lx,x,y)
        double precision x(lx),y(lx)
        do i=1,lx
          y(i)=x(i)
        end do
      return
      end

C---------------------------------------------------------------------------
C        zero - form zero array (Robinson, 1967)
C---------------------------------------------------------------------------
C        Input:
C          lx       length of x[]
C          x[]      array of polynomial coefficients
C
C        Output:
```

```
C          x[]      array of all zero values
C-----------------------------------------------------------------------------
      subroutine zero(lx,x)
         double precision x(lx)
         if (lx.le.0) return
         do i=1,lx
           x(i)=0.0
         end do
      return
      end

C-----------------------------------------------------------------------------
C        onelayer - calculates reflections using asymptotic formulas
C                       of Biot for one layer of medium (reference:
C                       Silin & Goloshubin, 2008).
C-----------------------------------------------------------------------------
C        Input:
C         kg       grain bulk modulus
C         rhog    grain density
C         kdry    dry rock bulk modulus
C         udry    dry rock shear modulus
C         phi      porosity
C         kappa  permeability
C         kf        fluid bulk modulus
C         rhof     fluid density
C         nf        fluid viscosity
C         freq     frequency
C         dt        sample rate or P wave time thickness of medium [1]
C
C        Output:
C         c1[]   array of coefficients in propagator matrix M
C         c2       coefficients outside the propagator matrix M
C         tau      slow wave time delay factor, tau = vfast / vslow
C-----------------------------------------------------------------------------
C        Note: Wave propagates from medium [1] to medium [2] as down going,
C                  from [2] to [1] as up going.
C-----------------------------------------------------------------------------
      subroutine onelayer(kg,rhog,kdry,udry,phi,kappa,kf,rhof,nf,
     +  freq,dt,c1,c2,tau)
         double precision kg(2),rhog(2),kdry(2),udry(2),phi(2),kappa(2)
         double precision kf(2),rhof(2),nf(2)
         double precision ksg_1,ksg_2,ksg_1up,ksg_2up,kfg_1,kfg_2
         double precision kfg_1up,kfg_2up
```

```
      double precision ks_1,ks_2,ks_1up,ks_2up,m_1,m_2,m_1up,m_2up
      double precision c1(4,4),c2
      integer tau

C     Input parameters

      pi=3.141592654
      w=2*pi*freq

C     Calculation for down going wave

      m_1=kdry(1)+udry(1)*4/3
      m_2=kdry(2)+udry(2)*4/3
      bf_1=1/kf(1)
      bf_2=1/kf(2)
      rhobulk_1=phi(1)*rhof(1)+(1-phi(1))*rhog(1)
      rhobulk_2=phi(2)*rhof(2)+(1-phi(2))*rhog(2)
      gk=rhof(2)*nf(1)*kappa(2)/(rhof(1)*nf(2)*kappa(1))
      gro_1=rhobulk_1/rhof(1)
      gro_2=rhobulk_2/rhof(2)
      e=0.000000986923*rhof(2)*kappa(2)*w/nf(2)
      vb_1=1000*sqrt(m_1/rhobulk_1)
      vb_2=1000*sqrt(m_2/rhobulk_2)
      vf_1=1000*sqrt(m_1/rhof(1))
      vf_2=1000*sqrt(m_2/rhof(2))
      ksg_1=kg(1)/(1-phi(1))
      ksg_2=kg(2)/(1-phi(2))
      kfg_1=kg(1)/(1-kdry(1)/kg(1))
      kfg_2=kg(2)/(1-kdry(2)/kg(2))
      gb_1=m_1*(bf_1*phi(1)+(1-phi(1))/kfg_1)
      gb_2=m_2*(bf_2*phi(2)+(1-phi(2))/kfg_2)
      gm_1=1-(1-phi(1))*kdry(1)/ksg_1
      gm_2=1-(1-phi(2))*kdry(2)/ksg_2
      z_1=10**6*m_1*sqrt((gb_1+gm_1**2)/gb_1)/vb_1
      z_2=10**6*m_2*sqrt((gb_2+gm_2**2)/gb_2)/vb_2
      a=(gm_1/(gm_1**2+gb_1)-gm_2/(gm_2**2+gb_2))*2*z_1*z_2/(z_1+z_2)
      d=0.000001*z_1*z_2*sqrt(gb_1*gb_2)*(vb_1*sqrt(gm_1**2+gb_1)/
     $  (sqrt(gk)*gro_2*m_1)+vb_2*sqrt(gm_2**2+gb_2)/(gro_1*m_2))/
     $ (gm_1*gm_2)

C     Calculation for up going wave

      m_1up=kdry(2)+udry(2)*4/3
```

```
      m_2up=kdry(1)+udry(1)*4/3
      bf_1up=1/kf(2)
      bf_2up=1/kf(1)
      rhobulk_1up=phi(2)*rhof(2)+(1-phi(2))*rhog(2)
      rhobulk_2up=phi(1)*rhof(1)+(1-phi(1))*rhog(1)
      gkup=rhof(1)*nf(2)*kappa(1)/(rhof(2)*nf(1)*kappa(2))
      gro_1up=rhobulk_1up/rhof(2)
      gro_2up=rhobulk_2up/rhof(1)
      eup=0.000000986923*rhof(1)*kappa(1)*w/nf(1)
      vb_1up=1000*sqrt(m_1up/rhobulk_1up)
      vb_2up=1000*sqrt(m_2up/rhobulk_2up)
      vf_1up=1000*sqrt(m_1up/rhof(2))
      vf_2up=1000*sqrt(m_2up/rhof(1))
      ksg_1up=kg(2)/(1-phi(2))
      ksg_2up=kg(1)/(1-phi(1))
      kfg_1up=kg(2)/(1-kdry(2)/kg(2))
      kfg_2up=kg(1)/(1-kdry(1)/kg(1))
      gb_1up=m_1up*(bf_1up*phi(2)+(1-phi(2))/kfg_1up)
      gb_2up=m_2up*(bf_2up*phi(1)+(1-phi(1))/kfg_2up)
      gm_1up=1-(1-phi(2))*kdry(2)/ksg_1up
      gm_2up=1-(1-phi(1))*kdry(1)/ksg_2up
      z_1up=10**6*m_1up*sqrt((gb_1up+gm_1up**2)/gb_1up)/vb_1up
      z_2up=10**6*m_2up*sqrt((gb_2up+gm_2up**2)/gb_2up)/vb_2up
      aup=(gm_1up/(gm_1up**2+gb_1up)-gm_2up/(gm_2up**2+gb_2up))*
     $ 2*z_1up*z_2up/(z_1up+z_2up)
      dup=0.000001*z_1up*z_2up*sqrt(gb_1up*gb_2up)*
     $ (vb_1up*sqrt(gm_1up**2+gb_1up)/(sqrt(gkup)*gro_2up*m_1up)+
     $ vb_2up*sqrt(gm_2up**2+gb_2up)/(gro_1up*m_2up))/(gm_1up*gm_2up)

C     Reflection coefficients and transmission coefficients

C     For fast incident and down going wave

      r_1fs=(gm_2**2+gb_2)*a/(gm_2*d)
      t_1fs=(gm_1**2+gb_1)*a/(gm_1*d)
      r_1ff=z_2*(t_1fs-r_1fs)/(z_1+z_2)
      t_1ff=z_1*(r_1fs-t_1fs)/(z_1+z_2)
      rff=(z_1-z_2)/(z_1+z_2)+sqrt(e)*r_1ff/2**0.5
      rffi=sqrt(e)*r_1ff/2**0.5
      tff=1+(z_1-z_2)/(z_1+z_2)+sqrt(e)*t_1ff/2**0.5
      tffi=sqrt(e)*t_1ff/2**0.5
      rfs=r_1fs*sqrt(e)/2**0.5
      rfsi=r_1fs*sqrt(e)/2**0.5
```

```
      tfs=t_1fs*sqrt(e)/2**0.5
      tfsi=t_1fs*sqrt(e)/2**0.5

C     For fast incident and up going wave

      r_1fsup=(gm_2up**2+gb_2up)*aup/(gm_2up*dup)
      t_1fsup=(gm_1up**2+gb_1up)*aup/(gm_1up*dup)
      r_1ffup=z_2up*(t_1fsup-r_1fsup)/(z_1up+z_2up)
      t_1ffup=z_1up*(r_1fsup-t_1fsup)/(z_1up+z_2up)
      rffup=(z_1up-z_2up)/(z_1up+z_2up)+sqrt(eup)*r_1ffup/2**0.5
      rffiup=sqrt(eup)*r_1ffup/2**0.5
      tffup=1+(z_1up-z_2up)/(z_1up+z_2up)+sqrt(eup)*t_1ffup/2**0.5
      tffiup=sqrt(eup)*t_1ffup/2**0.5
      rfsup=r_1fsup*sqrt(eup)/2**0.5
      rfsiup=r_1fsup*sqrt(eup)/2**0.5
      tfsup=t_1fsup*sqrt(eup)/2**0.5
      tfsiup=t_1fsup*sqrt(eup)/2**0.5

C     For slow incident and down going wave

      cs_1=gm_1+gb_1/gm_1
      cs_2=gm_2+gb_2/gm_2
      xs_1=-1/gm_1
      xs_2=-1/gm_2
      ks_1=(1/vf_1)*sqrt(gb_1+gm_1**2)
      ks_2=(1/vf_2)*sqrt(gb_2+gm_2**2)
      rss=(-cs_1*m_2*ks_2*xs_2/sqrt(gk)+cs_2*m_1*ks_1*xs_1)/
     $ (cs_1*m_2*ks_2*xs_2/sqrt(gk)+cs_2*m_1*ks_1*xs_1)
      tss=(cs_2*m_2*ks_2*xs_2/sqrt(gk)+cs_1*m_1*ks_1*xs_1)/
     $ (cs_1*m_2*ks_2*xs_2/sqrt(gk)+cs_2*m_1*ks_1*xs_1)
      rsf=z_2*(-1-rss+tss)/(z_1+z_2)
      tsf=-z_1*(-1-rss+tss)/(z_1+z_2)

C     For slow incident and up going wave

      cs_1up=gm_1up+gb_1up/gm_1up
      cs_2up=gm_2up+gb_2up/gm_2up
      xs_1up=-1/gm_1up
      xs_2up=-1/gm_2up
      ks_1up=(1/vf_1up)*sqrt(gb_1up+gm_1up**2)
      ks_2up=(1/vf_2up)*sqrt(gb_2up+gm_2up**2)
      rssup=(-cs_1up*m_2up*ks_2up*xs_2up/sqrt(gkup)+
     $ cs_2up*m_1up*ks_1up*xs_1up)/(cs_1up*m_2up*ks_2up*
```

```
     $    xs_2up/sqrt(gkup)+cs_2up*m_1up*ks_1up*xs_1up)
          tssup=(cs_2up*m_2up*ks_2up*xs_2up/sqrt(gkup)+cs_1up*
     $    m_1up*ks_1up*xs_1up)/(cs_1up*m_2up*ks_2up*xs_2up/
     $    sqrt(gkup)+cs_2up*m_1up*ks_1up*xs_1up)
          rsfup=z_2up*(-1-rssup+tssup)/(z_1up+z_2up)
          tsfup=-z_1up*(-1-rssup+tssup)/(z_1up+z_2up)

C         Velocity and attenuation coefficient

          vfast_1=vb_1*sqrt(1+gm_1**2/gb_1)
          vfast_2=vb_2*sqrt(1+gm_2**2/gb_2)
          vslow_1=vf_1*sqrt(2*eup/(gb_1+gm_1**2))
          vslow_2=vf_2*sqrt(2*e/(gb_2+gm_2**2))
          zeta_0_1=(gm_1**2+gb_1)/(gb_1*gro_1)
          zeta_0_2=(gm_2**2+gb_2)/(gb_2*gro_2)
          zeta_1_1=(((gm_1**2+gb_1)/gro_1)-gm_1)**2/(gb_1*(gm_1**2+gb_1))
          zeta_1_2=(((gm_2**2+gb_2)/gro_2)-gm_2)**2/(gb_2*(gm_2**2+gb_2))
          afast_1=w*sqrt(gb_1/(gb_1+gm_1**2))*zeta_1_1*eup/(vb_1*2*zeta_0_1)
          afast_2=w*sqrt(gb_2/(gb_2+gm_2**2))*zeta_1_2*e/(vb_2*2*zeta_0_2)
          aslow_1=w*sqrt((gb_1+gm_1**2)/(2*eup))/vf_1
          aslow_2=w*sqrt((gb_2+gm_2**2)/(2*e))/vf_2

C         For purely elastic media

          rff_elastic=(z_1-z_2)/(z_1+z_2)
          tff_elastic=1+(z_1-z_2)/(z_1+z_2)
          rffup_elastic=(z_1up-z_2up)/(z_1up+z_2up)
          tffup_elastic=1+(z_1up-z_2up)/(z_1up+z_2up)

C         Calculate the Matrix elements of M_0

          att_f=exp(-afast_1*vfast_1*dt)
          att_s=exp(-aslow_1*vfast_1*dt)

          c1(1,1)=(tff*(tffup*tssup-tsfup*tfsup)+rffup*
     $    (-tssup*rff+tsfup*rfs)+rsfup*(-rfs*tffup+rff*tfsup))*att_f
          c1(1,2)=(tsf*(tffup*tssup-tsfup*tfsup)+rffup*
     $    (-tssup*rsf+tsfup*rss)+rsfup*(-rss*tffup+rsf*tfsup))*att_s
          c1(1,3)=(rffup*tssup-rsfup*tfsup)/att_f
          c1(1,4)=(-rffup*tsfup+rsfup*tffup)/att_s
          c1(2,1)=(tfs*(tffup*tssup-tsfup*tfsup)+rfsup*
     $    (-tssup*rff+tsfup*rfs)+rssup*(-rfs*tffup+rff*tfsup))*att_f
          c1(2,2)=(tss*(tffup*tssup-tsfup*tfsup)+rfsup*
```

```
$    (-tssup*rsf+tsfup*rss)+rssup*(-rss*tffup+rsf*tfsup))*att_s
      c1(2,3)=(rfsup*tssup-rssup*tfsup)/att_f
      c1(2,4)=(-rfsup*tsfup+rssup*tffup)/att_s
      c1(3,1)=(-tssup*rff+tsfup*rfs)*att_f
      c1(3,2)=(-tssup*rsf+tsfup*rss)*att_s
      c1(3,3)=tssup/att_f
      c1(3,4)=-tsfup/att_s
      c1(4,1)=(-rfs*tffup+rff*tfsup)*att_f
      c1(4,2)=(-rss*tffup+rsf*tfsup)*att_s
      c1(4,3)=-tfsup/att_f
      c1(4,4)=tffup/att_s
      c2=tffup*tssup-tsfup*tfsup

      tau=vfast_1/vslow_1

 return
 end
```