Response of a Viscoelastic Layered Half-space to an Arbitrary 3-D Vector Body Force

Following our previous derivation for a response of a homogeneous elastic half-space to an arbitrary 3-D vector body force [*Smith and Sandwell*, 2003], we now wish to extend our Fourier treatment to that of a *layered elastic half-space*. Most of this formulation will follow the steps taken by *Rundle and Jackson* [1977], where they first review the 2-D case of an infinite elastic medium of two rigidities, μ_1, and μ_2 [*Ishii and Takagi*, 1967, *Rybicki*, 1971], followed by the 2-D case for a medium consisting of an elastic layer overlying an elastic half-space [*Rybicki*, 1971]. Their final step is to take the "form" of the 2-D layered elastic solution and apply it to the exact 3-D analytic solution for a double-couple source [*Steketee*, 1958]. Our development will follow these three steps, with the exception that our solutions will be prepared in the Fourier domain. The method follows:

i.) Develop the body force solution in the Fourier domain for a **2-D line-source** in a **homogenous** (μ) elastic **full-space**.

ii.) Extend the 2-D Fourier solution for a **homogeneous full-space** to that of a **full-space** of **varying rigidities** (μ_1 and μ_2) by comparing the 2-D Fourier homogeneous solution to the 2-D *Rundle and Jackson* [1977] solution for an infinite elastic medium of rigidities, μ_1 and μ_2.

iii.) Reduce the 2-D Fourier **full-space** solution of varying rigidities (μ_1 and μ_2) to that of a **layered half-space** (μ_1 and μ_2) through the method of images [*Rybicki*, 1971, *Rundle and Jackson*, 1977].

iv.) Formulate the **3-D** Fourier layered half-space solution by combining the **2-D** Fourier **layered** half-space solution with the "form" of the **3-D** Fourier **homogeneous** half-space solution [*Smith and Sandwell*, 2003].

v.) Identify the compact form of the 3-D Fourier layered half-space solution by showing that the layered half-space solution is easily obtained from the 3-D Fourier homogeneous half-space solution by a **substitution** of the **(z-a)** terms (and those like it) for new **(z-a-2mH)** terms.

vi.) 2-D analytic test for a line-source.

vii.) **Integrate** the 3-D Fourier layered half-space solution for a line-source over depth to obtain the **fault-plane solution**; identify the compact form of 3-D fault-plane solution.

viii.) Analytic treatment of an infinite sum

ix.) 2-D analytic test for a fault-plane.

x.) Modify the **Boussinesq** solution used in the homogeneous half-space solution to reflect **Burmister'**s equations for normal tractions in a layered half-space.

xi.) Incorporate time-dependence: the Maxwell model

xii.) Fortran source code for viscoelastic layered model

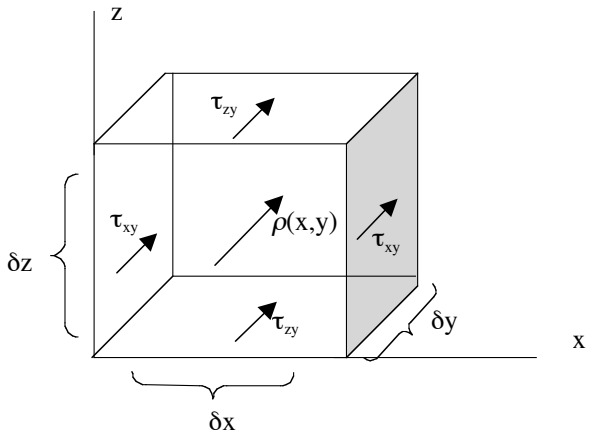
i) ***Develop the body force solution in the Fourier domain for a 2-D line-source in a homogenous (μ) elastic full-space.***

This section will consist of three steps in order to obtain the Green's function for a 2-D line-source:

(a) Force balance of an infinitely-long line-source
(b) 2-D Fourier transform of force balance
(c) Complex integration for the inverse Fourier transform in the z-direction

(a) Force balance of an infinitely-long line-source:

Consider the forces acting on an infinitely-long square rod depicted below. The body force per unit volume of rod must be balanced by tractions on the sides of the rod.



The equation for this force balance is

$$\left[\tau_{xy}(x+\delta x)-\tau_{xy}(x)\right]\delta y\delta z+\left[\tau_{zy}(z+\delta z)-\tau_{zy}(z)\right]\delta x\delta y=-\rho(x,z)dxdydz \qquad (1)$$

$$\frac{\partial\tau_{xy}}{\partial x}+\frac{\partial\tau_{zy}}{\partial x}=-\rho(x,z) \qquad \text{with } \tau_{xy}=\mu\frac{\partial v}{\partial x} \text{ and } \tau_{zy}=\mu\frac{\partial v}{\partial z}$$

$$\mu\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial z^2}\right)=-\rho(x,z) \qquad \text{where } \mu = \text{shear modulus, v = y-displacement} \qquad (2)$$

(b) 2-D Fourier transform of force balance:

We can generate the solution to obtain an arbitrary source distribution by first developing the line-source Green's function. Consider a line-source at a depth of $z = a$. The differential equation (2) then becomes:

$$\mu\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial z^2}\right)=-\rho(x,z)=-F\delta(x)\delta(z-a)\,* \tag{3}$$

*note: an image-source at depth $z = -a$ can be introduced to reduce the full-space derivation to a half-space derivation. For this case, $F\delta(x)\delta(z+a)$ would be added to the terms on the right side.

By taking the 2-D Fourier transform of the above differential equation (3), we have

$$-(2\pi)^2\mu\left(k_x^2+k_z^2\right)V(\mathbf{k})=-F_y\left(e^{-2\pi i k_z a}\right)$$

or

$$V(\mathbf{k})=\frac{\left(e^{-2\pi i k_z a}\right)}{(2\pi)^2\mu\left(k_x^2+k_z^2\right)}F_y. \tag{4}$$

(c) Complex integration for the inverse Fourier transform in the z-direction

We next need to take the inverse Fourier transform of (4) with respect to k_z and make sure the solution goes to zero as |z| goes to infinity. The integral takes on the form:

$$V(k_x,z)=\frac{F_y}{(2\pi)^2\mu}\int_{-\infty}^{\infty}\frac{e^{i2\pi k_z(z-a)}}{\left(k_x^2+k_z^2\right)}dk_z=\frac{F_y}{(2\pi)^2\mu}\int_{-\infty}^{\infty}\frac{e^{i2\pi k_z(z-a)}}{\left(k_z+ik_x\right)^2\left(k_z-ik_x\right)^2}dk_z \tag{5}$$

From the Cauchy integral formula, we know that for any analytic function the following holds for a counterclockwise path surrounding the pole:

$$\oint\frac{f(z)}{z-z_0}dz=i2\pi f(z_0).$$

It is necessary to evaluate all cases for $k_x > 0$, $z > -a$ and $z < -a$, and for $k_x < 0$, $z > -a$ and $z < -a$. The result of (5) for all cases is simply

$$V(k_x,z) = -\left(\frac{e^{-2\pi k_x (z-a)}}{4\pi\mu|k_x|}\right)F_y \qquad \text{source only} \quad \text{(full - space)} \tag{6}$$

or

$$V(k_x,z) = -\left(\frac{e^{-2\pi k_x (z-a)}}{4\pi\mu|k_x|} + \frac{e^{-2\pi k_x (-z-a)}}{4\pi\mu|k_x|}\right)F_y \quad \text{source + image} \quad \text{(half - space).}$$

By taking the inverse cosine transform of (6), we have the Green's function in the form of *ln(r)*:

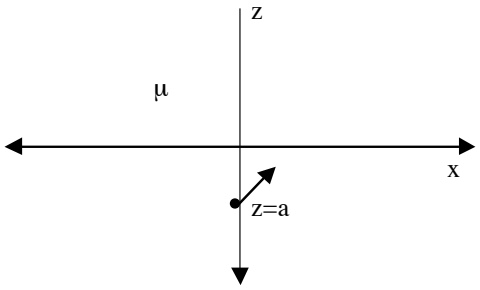
$$v(x,z) = -\frac{F_y}{2\pi\mu}\left(\ln\left[x^2 + (z-a)^2\right]^{1/2}\right) \quad \text{for a full - space} \tag{7}$$

or

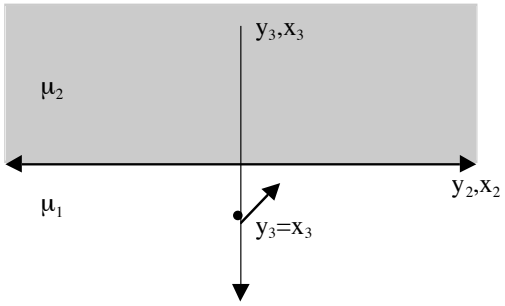
$$v(x,z) = -\frac{F_y}{2\pi\mu}\left(\ln\left[x^2 + (z-a)^2\right]^{1/2} + \ln\left[x^2 + (-z-a)^2\right]^{1/2}\right) \quad \text{for a half - space.}$$

ii) ***Extend the 2-D Fourier solution for a homogeneous full-space to that of a full-space of varying rigidities (μ_1 and μ_2) by comparing the 2-D Fourier homogeneous solution to the 2-D Rundle and Jackson [1977] solution for an infinite elastic medium of rigidities, μ_1 and μ_2.***

We will now compare our homogeneous full-space 2-D Fourier solution (6, 7) with the solution of *Rundle and Jackson* [1977] for an infinite elastic medium of rigidities μ_1 and μ_2.



2-D homogeneous full-space with line-source at $z = a$, rigidity μ.
(current Fourier model)



2-D full-space with line-source at $y_3=x_3$, rigidities μ_1 and μ_2
(Rundle and Jackson model*)

*x_3 = source location
y_3 = observation plane

Our Fourier full-space solution (7) looks like:

$$v(x,z) = -\frac{F_y}{2\pi\mu}\left(\ln\left[x^2 + (z-a)^2\right]^{1/2}\right). \qquad (7)$$

The *Rundle and Jackson* full-space solution (eq. 1) for $y_3 \geq 0$, $x_3 \geq 0$ is:

$$K_1^1(y,x) = -\frac{1}{2\pi\mu_1}\left\{\log\left[\left(y_2 - x_2\right)^2 + \left(y_3 - x_3\right)^2\right]^{1/2} + \frac{\mu_1 - \mu_2}{\mu_1 + \mu_2}\log\left[\left(y_2 - x_2\right)^2 + \left(y_3 + x_3\right)^2\right]^{1/2}\right\}$$

By making the following substitutions, we can directly compare our Fourier solution. Let $y_2 = x$, $y_3 = z$, $x_2 = 0$ (source at origin), $x_3 = a$ (source depth), and $\mu_1 = \mu_2 = 1$.

In doing so, we find that the *Rundle and Jackson* solution (8) is reduced to:

$$K_1^1 = -\frac{1}{2\pi\mu}\left(\log\left[x^2 + (z-a)^2\right]^{1/2}\right), \qquad (9)$$

which is identical to our Fourier solution (7)!

Our next step is to use this information to "back out" a term to include in our Fourier solution that accounts for a medium of different rigidities.

For a full-space, our Fourier solution (7) can be modified to look like the *Rundle and Jackson* solution (8) by adding a similar 2nd term with the coefficient $\frac{\mu_1 - \mu_2}{\mu_1 + \mu_2}$:

$$v(x,z) = -\frac{F_y}{2\pi\mu}\left(\log\left[x^2 + (z-a)^2\right]^{1/2} + \frac{\mu_1 - \mu_2}{\mu_1 + \mu_2}\log\left[x^2 + (z+a)^2\right]^{1/2}\right). \tag{10}$$

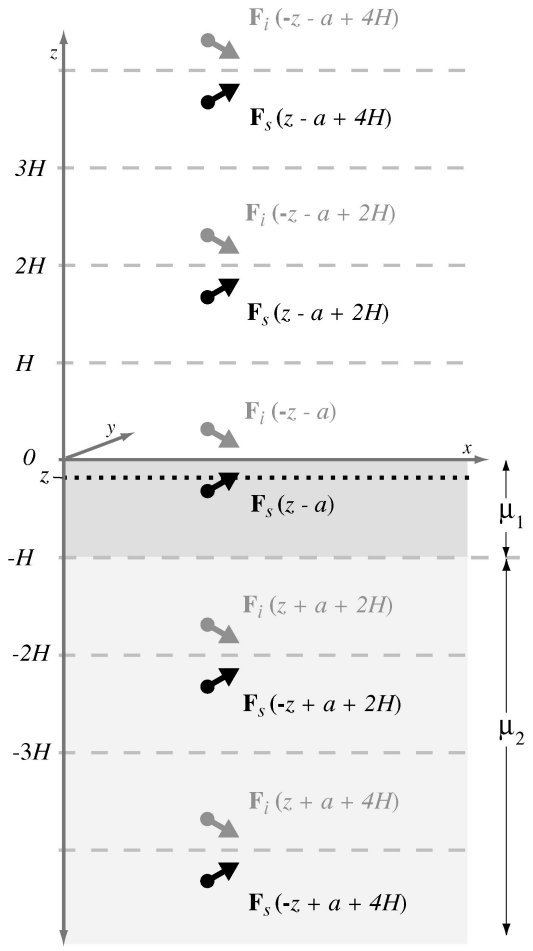
In the Fourier domain, this solution looks like

$$V(k_x,z) = -\frac{1}{4\pi\mu|k_x|}\left(e^{-2\pi k_x(z-a)} + \frac{\mu_1 - \mu_2}{\mu_1 + \mu_2}e^{-2\pi k_x(-z-a)}\right)F_y. \tag{11}$$

This is the Fourier solution for an elastic full-space of varying rigidities. Extending this solution to that of a half-space is not quite as trivial as in our previous homogeneous half-space derivation, where a simple addition of an "image" source was all that was required for canceling shear stress at the surface and reducing the system to a half-space. We next refer to the *Rybicki* [1971] solution to the layered half-space, where multiple image sources are required for the reduction of the solution from a full-space to a half-space.

iii) ***Reduce the 2-D Fourier full-space solution of varying rigidities (μ_1 and μ_2) to that of a layered half-space (μ_1 and μ_2) through the method of images [Rybicki, 1971, Rundle and Jackson, 1977]***

We would now like to find the 2-D Fourier solution for a line-source in a layered half-space. From the method of images [*Weertman*, 1964], we know that we can place an image source opposite the original source to satisfy the requirements of a stress-free surface. For a layered half-space though, we must also require the displacements and consequent stresses to remain continuous at the boundary layer between rigidities. To do this, we must superpose multiple image sources, reflected both above and below the x-axis to account for both the source and the layer (H).



An infinite number of secondary source-image terms are, in turn, added to the solution. *Rundle and Jackson* (eq. 3) present a layered half-space solution for $y_3 > 0$, $x_3 > 0$ as:

$$G_1^1(y,x) = -\frac{1}{2\pi\mu_1}\left\{\begin{array}{l}\log\left[\left(y_2-x_2\right)^2+\left(y_3-x_3\right)^2\right]^{1/2}+\log\left[\left(y_2-x_2\right)^2+\left(y_3+x_3\right)^2\right]^{1/2} \\ +\sum_{m=1}^{\infty}\left(\frac{\mu_1-\mu_2}{\mu_1+\mu_2}\right)^m\left[\begin{array}{l}\log\left[\left(y_2-x_2\right)^2+\left(y_3-x_3-2mH\right)^2\right]^{1/2} \\ +\log\left[\left(y_2-x_2\right)^2+\left(y_3+x_3-2mH\right)^2\right]^{1/2} \\ +\log\left[\left(y_2-x_2\right)^2+\left(y_3-x_3+2mH\right)^2\right]^{1/2} \\ +\log\left[\left(y_2-x_2\right)^2+\left(y_3+x_3+2mH\right)^2\right]^{1/2}\end{array}\right]\end{array}\right\}, \tag{12}$$

which forms the source-image combination of

$$G_1^1(y,x) = -\frac{1}{2\pi\mu_1}\left\{\begin{array}{l}\text{source} + \text{image1} \\ +\sum_{m=1}^{\infty}\left(\frac{\mu_1-\mu_2}{\mu_1+\mu_2}\right)^m\left[\begin{array}{l}\text{image2} \\ +\ \text{image3} \\ +\ \text{image4} \\ +\ \text{image5}\end{array}\right]\end{array}\right\}. \tag{13}$$

Following this concept, we can modify our full-space layered solution with the addition of similar terms. For our solution, we shall call all terms placed below the x-axis ($z = a$) "source" terms, with the secondary terms called source1 and source2. Those placed above the x-axis ($z = - a$) will be called "image" terms, again with the secondary terms called image1 and image2. With this in mind, our layered half-space solution should look like:

$$V(k_x,z) = -\frac{F_y}{4\pi\mu_1|k_x|}\left(e^{-2\pi k_x(z-a)}+e^{-2\pi k_x(-z-a)}+\sum_{m=1}^{\infty}\left(\frac{\mu_1-\mu_2}{\mu_1+\mu_2}\right)^m\left[\begin{array}{l}e^{-2\pi k_x(-z+a+2mH)} \\ +e^{-2\pi k_x(-z-a+2mH)} \\ +e^{-2\pi k_x(z-a+2mH)} \\ +e^{-2\pi k_x(+z+a+2mH)}\end{array}\right]\right) \tag{14}$$

which forms the source-image combination of

$$V(k_x,z) = -\frac{F_y}{4\pi\mu_1|k_x|}\left\{\begin{array}{l}\text{source} + \text{image} \\ +\sum_{m=1}^{\infty}\left(\frac{\mu_1-\mu_2}{\mu_1+\mu_2}\right)^m\left[\begin{array}{l}\text{source1} \\ +\ \text{image1} \\ +\ \text{source2} \\ +\ \text{image2}\end{array}\right]\end{array}\right\}. \tag{15}$$

---

iv) ***Formulate the 3-D Fourier layered half-space solution by combining the Fourier layered half-space solution with the "form" of the 3-D Fourier homogeneous half-space solution [Smith and Sandwell, 2003]***

The next step taken by *Rundle and Jackson* is to use the form of the exact 3-D analytic solution for a homogeneous half-space [*Stektee*, 1958] to find the 3-D solution for a layered half-space.

They combine *Stektee's* 3-D homogeneous solution for displacement in the y-direction (eq. 4):

$$U_{12}^{1}(x,y) = \frac{U_o dS}{8\pi}\left[2(1-\alpha)\frac{x_2}{R^3} + 6\alpha\frac{x_1^2 x_2}{R^5}\right] \quad \text{with } \alpha = \frac{\lambda+\mu}{\lambda+2\mu} \quad R = \left[x_1^2 + x_2^2 + (x_3 - y_3)^2\right]^{1/2}$$

and the "form" of their 2-D layered solution (12):

$$G_1^1(y,x) = -\frac{1}{2\pi\mu_1}\left\{\begin{array}{l} \log\left[(y_2-x_2)^2 + (y_3-x_3)^2\right]^{1/2} + \log\left[(y_2-x_2)^2 + (y_3+x_3)^2\right]^{1/2} \\ +\sum_{m=1}^{\infty}\left(\frac{\mu_1-\mu_2}{\mu_1+\mu_2}\right)^m \left[\begin{array}{l} \log\left[(y_2-x_2)^2 + (y_3-x_3-2mH)^2\right]^{1/2} \\ +\log\left[(y_2-x_2)^2 + (y_3+x_3-2mH)^2\right]^{1/2} \\ +\log\left[(y_2-x_2)^2 + (y_3-x_3+2mH)^2\right]^{1/2} \\ +\log\left[(y_2-x_2)^2 + (y_3+x_3+2mH)^2\right]^{1/2} \end{array}\right] \end{array}\right\}$$

to arrive at a preliminary solution for a 3-D layered half-space (eq. 5):

$$W_{12}^{1}(x,y) = \frac{U_o dS}{8\pi}\left[\begin{array}{l} 2(1-\alpha)\frac{x_2}{R^3} + 6\alpha\frac{x_1^2 x_2}{R^5} + 2(1-\alpha)\frac{x_2}{S^3} + 6\alpha\frac{x_1^2 x_2}{S^5} \\ +\sum_{m=1}^{\infty}\left(\frac{\mu_1-\mu_2}{\mu_1+\mu_2}\right)^m \left[\begin{array}{l} 2(1-\alpha)\frac{x_2}{R_{--}^3} + 6\alpha\frac{x_1^2 x_2}{R_{--}^5} \\ +2(1-\alpha)\frac{x_2}{R_{-+}^3} + 6\alpha\frac{x_1^2 x_2}{R_{-+}^5} \\ +2(1-\alpha)\frac{x_2}{R_{+-}^3} + 6\alpha\frac{x_1^2 x_2}{R_{+-}^5} \\ +2(1-\alpha)\frac{x_2}{R_{++}^3} + 6\alpha\frac{x_1^2 x_2}{R_{++}^5} \end{array}\right] \end{array}\right]$$

with

$$R = \left[x_1^2 + x_2^2 + (x_3 - y_3)^2\right]^{1/2}$$
$$S = \left[x_1^2 + x_2^2 + (x_3 + y_3)^2\right]^{1/2}$$
$$R_{--} = \left[x_1^2 + x_2^2 + (x_3 - y_3 - 2mH)^2\right]^{1/2}$$
$$R_{-+} = \left[x_1^2 + x_2^2 + (x_3 + y_3 - 2mH)^2\right]^{1/2}$$
$$R_{+-} = \left[x_1^2 + x_2^2 + (x_3 - y_3 + 2mH)^2\right]^{1/2}$$
$$R_{++} = \left[x_1^2 + x_2^2 + (x_3 + y_3 + 2mH)^2\right]^{1/2}.$$

We will use the same procedure by combining our 3-D Fourier homogeneous half-space solution [*Smith and Sandwell*, 2003] with the form of our 2-D Fourier layered half-space

solution (14). From our previous line-source derivation (p. 7, notes), we found the V_y component of displacement (source + image) to be

$$V_y(\mathbf{k},z) = \frac{C}{2\pi|\mathbf{k}|}\left\{e^{-2\pi|\mathbf{k}|(z-a)}\left[D + \frac{k_x^2}{|\mathbf{k}|^2} - 2\pi|\mathbf{k}|(z-a)\frac{k_y^2}{|\mathbf{k}|^2}\right] + e^{-2\pi|\mathbf{k}|(-z-a)}\left[D + \frac{k_x^2}{|\mathbf{k}|^2} - 2\pi|\mathbf{k}|(-z-a)\frac{k_y^2}{|\mathbf{k}|^2}\right]\right\}F_y \qquad (16)$$

We will now take each term from (16) and expand it about the 2-D layered half-space solution (14)

$$V(k_x,z) = -\frac{F_y}{4\pi\mu_1|k_x|}\left(e^{-2\pi k_x(z-a)} + e^{-2\pi k_x(-z-a)} + \sum_{m=1}^{\infty}\left(\frac{\mu_1-\mu_2}{\mu_1+\mu_2}\right)^m\begin{bmatrix} e^{-2\pi k_x(-z+a+2mH)} \\ +e^{-2\pi k_x(-z-a+2mH)} \\ +e^{-2\pi k_x(z-a+2mH)} \\ +e^{-2\pi k_x(+z+a+2mH)}\end{bmatrix}\right) .$$

We then have:

$$V_y^L(\mathbf{k},z) = \frac{C}{2\pi|\mathbf{k}|}\left\{\begin{array}{l} D\left[e^{-2\pi k_x(z-a)} + e^{-2\pi k_x(-z-a)} + \sum_{m=1}^{\infty}\left(\frac{\mu_1-\mu_2}{\mu_1+\mu_2}\right)^m\begin{bmatrix} e^{-2\pi k_x(-z+a+2mH)} \\ +e^{-2\pi k_x(-z-a+2mH)} \\ +e^{-2\pi k_x(z-a+2mH)} \\ +e^{-2\pi k_x(z+a+2mH)}\end{bmatrix}\right] \\ +\frac{k_x^2}{|\mathbf{k}|^2}\left[e^{-2\pi k_x(z-a)} + e^{-2\pi k_x(-z-a)} + \sum_{m=1}^{\infty}\left(\frac{\mu_1-\mu_2}{\mu_1+\mu_2}\right)^m\begin{bmatrix} e^{-2\pi k_x(-z+a+2mH)} \\ +e^{-2\pi k_x(-z-a+2mH)} \\ +e^{-2\pi k_x(z-a+2mH)} \\ +e^{-2\pi k_x(z+a+2mH)}\end{bmatrix}\right] \\ -2\pi|\mathbf{k}|\frac{k_y^2}{|\mathbf{k}|^2}\left[(z-a)e^{-2\pi k_x(z-a)} + (-z-a)e^{-2\pi k_x(-z-a)} + \sum_{m=1}^{\infty}\left(\frac{\mu_1-\mu_2}{\mu_1+\mu_2}\right)^m\begin{bmatrix} (z-a-2mH)e^{-2\pi k_x(-z+a+2mH)} \\ +(-z-a+2mH)e^{-2\pi k_x(-z-a+2mH)} \\ +(z-a+2mH)e^{-2\pi k_x(z-a+2mH)} \\ +(-z-a-2mH)e^{-2\pi k_x(z+a+2mH)}\end{bmatrix}\right] \end{array}\right\}F_y \qquad (17)$$

v) ***Identify the compact form of the 3-D Fourier layered half-space solution by showing that the layered half-space solution is easily obtained from the 3-D Fourier homogeneous half-space solution by a substitution of the (z-a) terms (and those like it) for new (z-a-2mH) terms.***

Close inspection of (16) and (17) will give rise to a simplification of the 2-D --> 3-D formulation. From (16), we see that homogeneous half-space solution consists of a source and image term that are functions of $(z - a)$ and $(-z - a)$:

$$V(\mathbf{k},z) = V_s(z-a) + V_i(-z-a) \tag{18}$$

where V_s = source component from homogeneous half - space solution
V_i = image component from homogeneous half - space solution

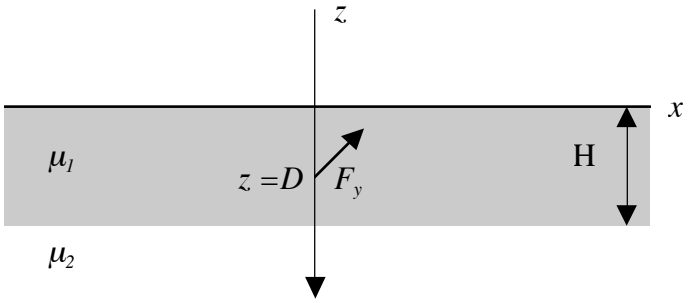
From (18), we see that the source-image combination of the 3-D half-space homogeneous solution can easily be extended to the form of (17) by:

$$V^L(\mathbf{k},z) = V_s^L(z-a) + V_i^L(-z-a) + \sum_{m=1}^{\infty}\left(\frac{\mu_1-\mu_2}{\mu_1+\mu_2}\right)^m \begin{bmatrix} V_s^L(z-a-2mH) \\ +V_i^L(-z-a+2mH) \\ +V_s^L(z-a+2mH) \\ +V_i^L(-z-a-2mH) \end{bmatrix} \tag{19}$$

where $V_{s,i}^L$ = homogenous solutions (p. 7, elastic notes) in the form of (17).

vii) ***2-D analytic test for a line-source***

We will now compare our Fourier-derived layered half-space solution with the analytic solution for a layered half-space.



The analytic solution (derived from *Rybicki's* fault-plane solution) is:

$$V = \frac{V_o}{\pi}\left[\frac{x}{x^2 + D^2} + \sum_{m=1}^{\infty}\left(\frac{\mu_1 - \mu_2}{\mu_1 + \mu_2}\right)^m \left\{\frac{x}{x^2 + (2mH + D)^2} - \frac{x}{x^2 + (-2mH + D)^2}\right\}\right] \qquad (20)$$

where V_o= slip
D = line-source depth
H = boundary between rigidities μ_1 and μ_2.

For the following combinations of Lame parameters, for a line source located at D = 10, layer H = 30, m=100, the analytic solution behaves according to Figure 1:

$\lambda = \mu_1 = \mu_2$	homogeneous half - space
$\lambda = \mu_1 = 1, \ \mu_2 = 0$	elastic plate over fluid half - space
$\lambda = \mu_1 = 1, \ \mu_2 = 3.25$	layered half - space

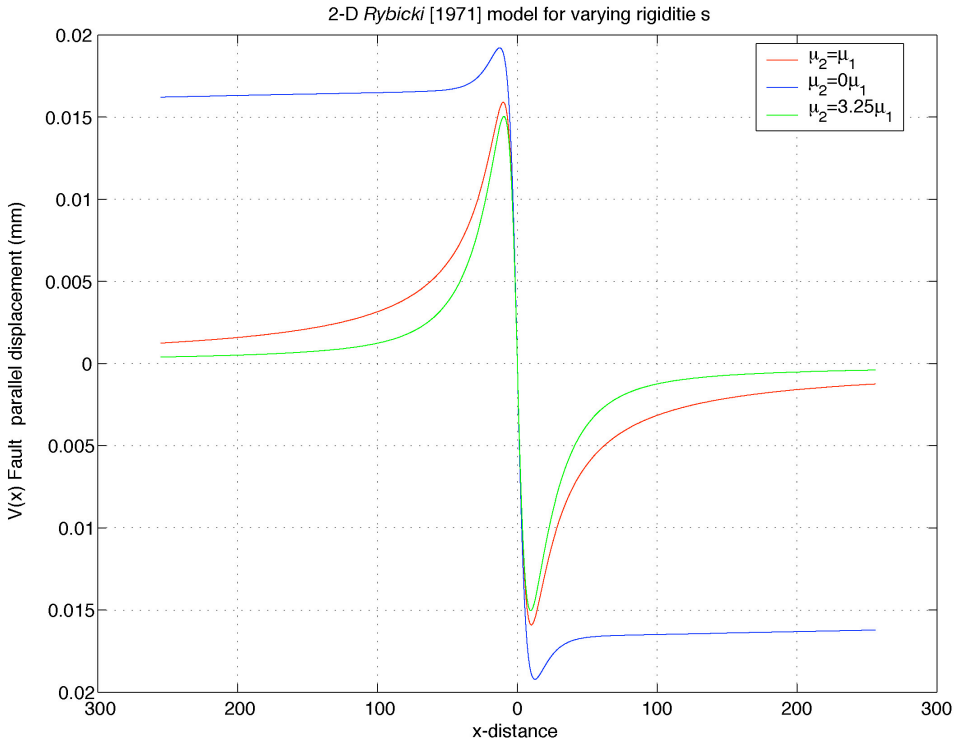


Figure 1. *Rybicki* [1971] models

We then compare our 3-D Fourier solution (20) for a layered half-space for an infinitely-long line-source to the 2-D analytic function above. In order to form a double-couple dislocation, we have constructed a screw dislocation by taking the derivative of the point source in the direction normal to the fault plane. This corresponds to multiplication in the Fourier transform domain. For this 2-D comparison, we have placed an infinitely long line source in the y-direction, and thus use the following derivative:

$$\Im\left[\frac{df(x,y)}{dx}\right] = i2\pi k_x F\left(k_x, k_y\right)$$

The line source is imbedded in a 2-D grid which is Fourier transformed, multiplied by the transfer functions above, and inverse Fourier transformed.

For comparison with the analytic solution of *Rybicki* (20), we must divide our Fourier solution by Δx which corresponds to the cell-size in the x-direction. As a first test, we set $\lambda = \mu_1 = \mu_2 = 1$ and test the case for a layered half-space of the same rigidity, better described as a homogeneous half-space. This numerical test will also be compared to both analytic solutions for a homogeneous half-space and for a layered half-space. The comparison between the three solutions is shown in Figure 1.

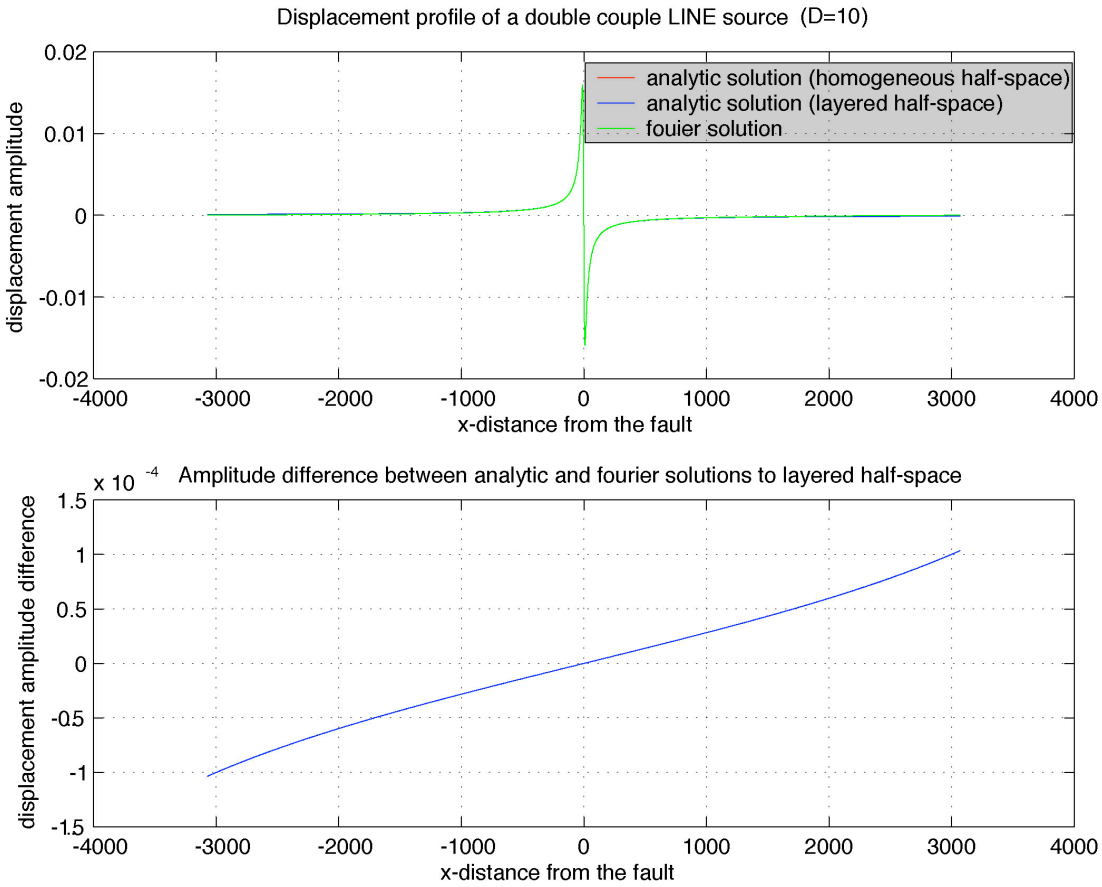


Figure 2. *Rybicki* [1971] and Fourier models

For a source depth of 10 km, the x-length of the Fourier grid must be greater than 6144 to achieve better than 10^{-4} numerical accuracy. A smaller dimension can be used if the fault has a finite length in the y-direction.

As a second test, we will inspect the case for a layered half-space with the half-space rigidity set to zero, simulating an elastic plate. We set $\lambda = \mu_1 = 1,\ \mu_2 = 0$ and and plot the results in Figure 3 for $m = 5$ summed terms.

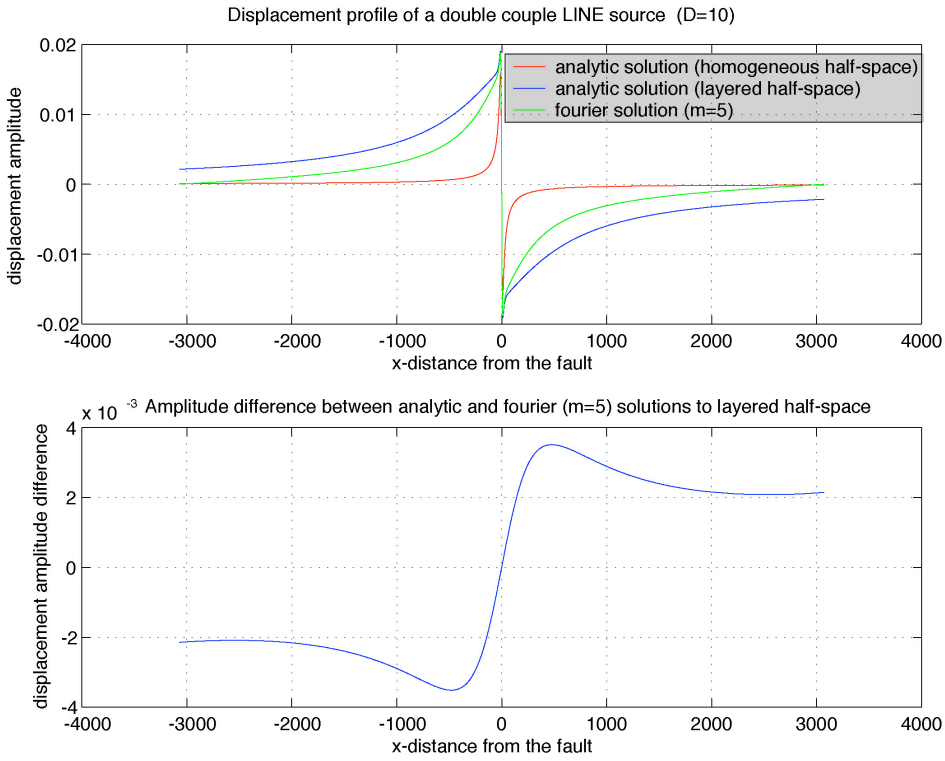


Figure 3. *Rybicki* [1971] and Fourier models for $m = 5$

From Figure 3, we note that both the analytical and Fourier solutions are limited by the number of m terms. Figure (4) shows how these solutions converge as $m \to \infty$ (analytic solution calculated for $m = 10$).

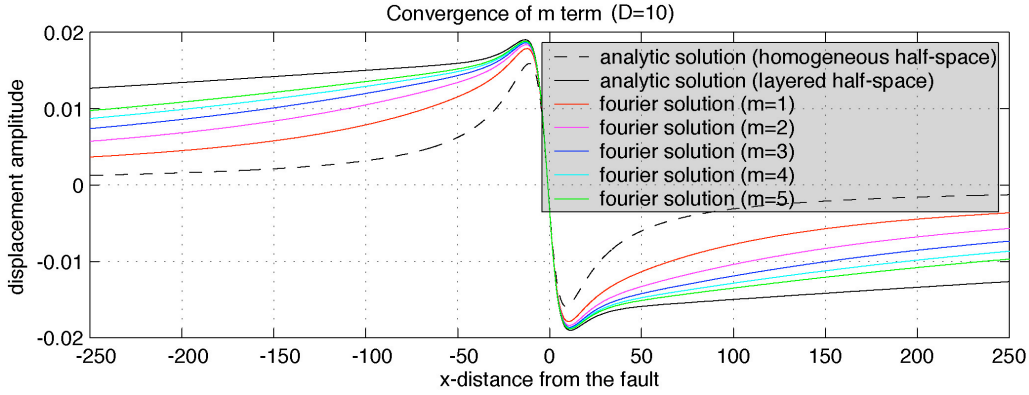


Figure 4. Convergence of Fourier models for various m.

For an elastic plate, $(\mu_1 - \mu_2 / \mu_1 + \mu_2)^m$ fluctuates between +/-1, and thus the solution depends on the inner +/- $2mH$ terms to converge.

As a third test, we will inspect the case for a layered half-space of varying rigidities. We set $\lambda = \mu_1 = 1$, $\mu_2 = 3.25$ and plot the results in Figure 5.

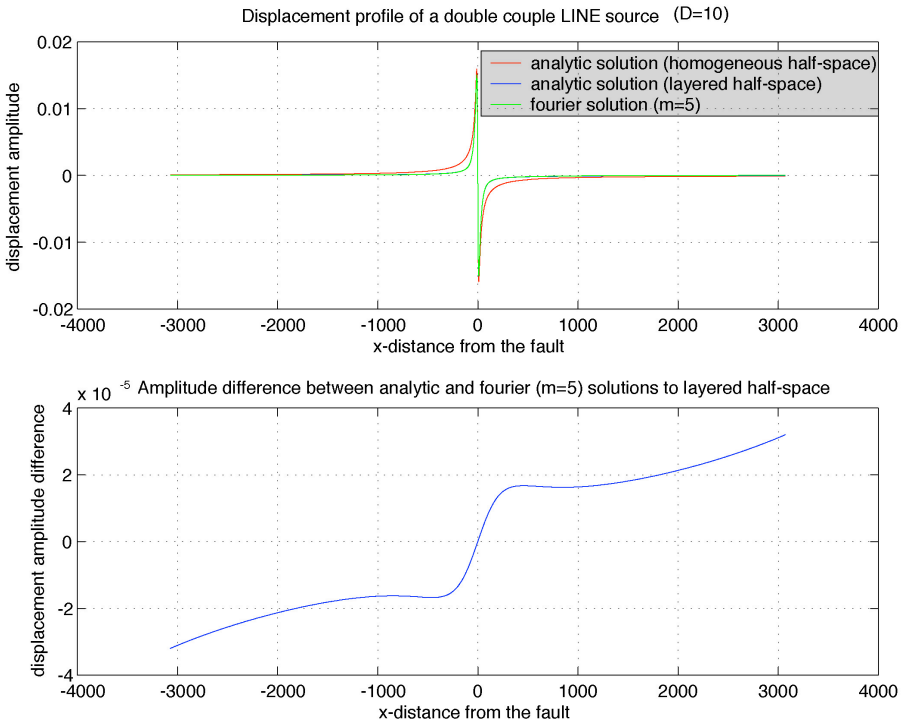


Figure 5. *Rybicki* [1971] and Fourier models for $\mu_2 = 3.25\mu_1$.

For a source depth of 10 km, the x-length of the Fourier grid must be greater than 6144 to achieve better than 10^{-5} numerical accuracy. A smaller dimension can be used if the fault has a finite length in the y-direction. In addition, Figure 5 shows our Fourier solution for $m = 5$ summed terms. Figure 6 shows how these solutions converge as $m \to \infty$.

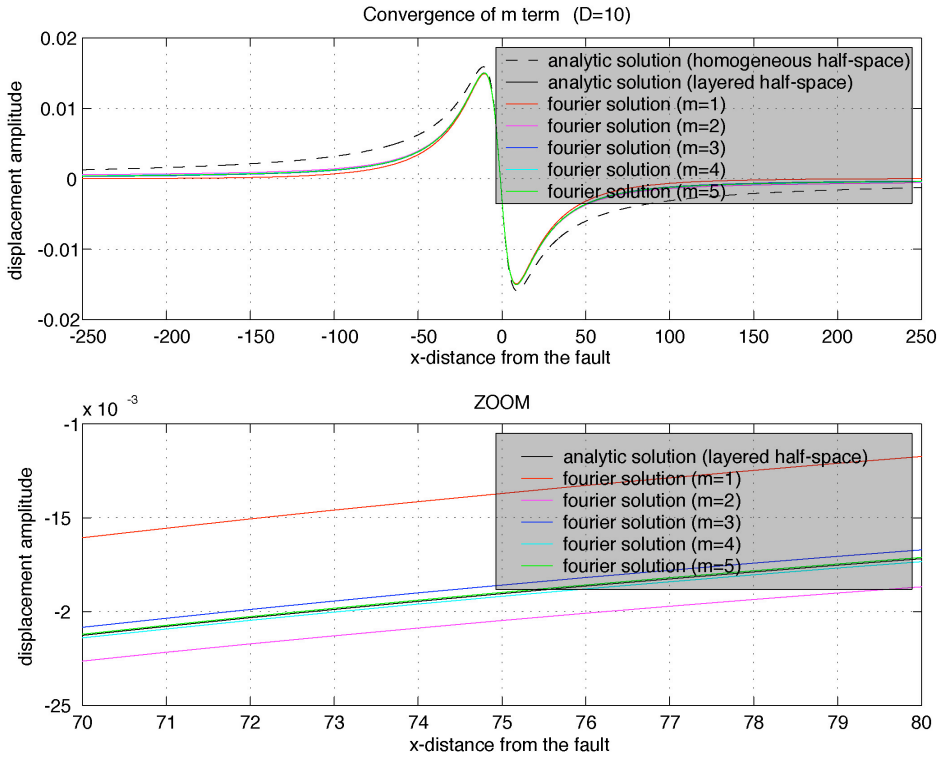


Figure 6. Convergence of Fourier solution for $\mu_2 = 3.25\mu_1$.

For the layered medium, $\mu_1 = 1,\ \mu_2 = 3.25$, $\left(\mu_1 - \mu_2 / \mu_1 + \mu_2\right)^m$ converges quickly, as demonstrated by Figure 6. The coefficient converges to zero for terms > m = 10.

vii) ***Integrate the 3-D Fourier layered half-space solution for a line-source over depth to obtain the fault-plane solution; identify the compact form of 3-D fault-plane solution.***

We next integrate the 3-D Fourier layered half-space solution (19) between depths d_1 and d_2 to simulate a fault plane. If the fault is assumed to be vertical, the integration can be performed analytically. We shall skip the major components of this integration, noting that the same exercise was discussed in the notes for a homogeneous half-space solution (p. 25). Here we take z' = a:

$$\int_{d_1}^{d_2} V^L(\mathbf{k},z)\ dz' \ = \ V^L_{fault} \tag{21}$$

Recall that the V_y component (source + image) of fault-plane solution for a homogeneous half-space looks like:

$$V_{fault,y}(\mathbf{k}) = \frac{C}{(2\pi\mathbf{k})^2}\left\{\begin{array}{l} e^{-2\pi\mathbf{k}(z-d_2)}\left[D+\frac{k_x^2}{|\mathbf{k}|^2}-\frac{k_y^2}{|\mathbf{k}|^2}\left(1+2\pi\mathbf{k}(z-d_2)\right)\right]-e^{-2\pi\mathbf{k}(z-d_1)}\left[D+\frac{k_x^2}{|\mathbf{k}|^2}-\frac{k_y^2}{|\mathbf{k}|^2}\left(1+2\pi\mathbf{k}(z-d_1)\right)\right] \\ +e^{-2\pi\mathbf{k}(-z-d_2)}\left[D+\frac{k_x^2}{|\mathbf{k}|^2}-\frac{k_y^2}{|\mathbf{k}|^2}\left(1+2\pi\mathbf{k}(-z-d_2)\right)\right]-e^{-2\pi\mathbf{k}(-z-d_1)}\left[D+\frac{k_x^2}{|\mathbf{k}|^2}-\frac{k_y^2}{|\mathbf{k}|^2}\left(1+2\pi\mathbf{k}(-z-d_1)\right)\right] \end{array}\right\} F_y \cdot \tag{22}$$

After integrating (21) over depth variable, z', we find the solution takes on the form of (22). The layered half-space solution for a fault plane is thus:

$$\begin{aligned} \int_{d_1}^{d_2} V^L(\mathbf{k},z)dz' = V^L_{fault} = &\left[V^L_{fault,s}(z-d_2)-V^L_{fault,s}(z-d_1)\right]+ \\ &\left[V^L_{fault,i}(-z-d_2)-V^L_{fault,i}(-z-d_1)\right]+ \\ &\sum_{m=1}^{\infty}\left(\frac{\mu_1-\mu_2}{\mu_1+\mu_2}\right)^m \left[\begin{array}{l} \left[V^L_{fault,s}(z-d_2-2mH)-V^L_{fault,s}(z-d_1-2mH)\right] \\ -\left[V^L_{fault,i}(-z-d_2+2mH)-V^L_{fault,i}(-z-d_1+2mH)\right] \\ +\left[V^L_{fault,s}(z-d_2+2mH)-V^L_{fault,s}(z-d_1+2mH)\right] \\ -\left[V^L_{fault,i}(-z-d_2-2mH)-V^L_{fault,i}(-z-d_1-2mH)\right] \end{array}\right] . \end{aligned} \tag{23}$$

Note that the integrated fault plane solution (23) looks similar to the form of the line-source solution (15)

$$V(k_x,z)=\left\{\begin{array}{c}\text{source}\quad+\quad\text{image}\\ +\sum_{m=1}^{\infty}\left(\frac{\mu_1-\mu_2}{\mu_1+\mu_2}\right)^m\begin{bmatrix}\text{source1}\\ +\ \text{image1}\\ +\ \text{source2}\\ +\ \text{image2}\end{bmatrix}\end{array}\right\}$$

with the exception (and a major one at that!) that image1 and image2 components are *subtracted* instead of added:

$$V^L_{fault}(\mathbf{k},z)=\left\{\begin{array}{c}\text{source}\quad+\quad\text{image}\\ +\sum_{m=1}^{\infty}\left(\frac{\mu_1-\mu_2}{\mu_1+\mu_2}\right)^m\begin{bmatrix}\text{source1}\\ -\ \text{image1}\\ +\ \text{source2}\\ -\ \text{image2}\end{bmatrix}\end{array}\right\}.$$

A general solution for all three displacement components can be represented by

$$\begin{aligned}\int_{d_1}^{d_2}\mathbf{U}(\mathbf{k},Z)dZ \;=\;& \left[\mathbf{U}'_s(z-d_2)-\mathbf{U}'_s(z-d_1)\right]+\left[\mathbf{U}'_i(-z-d_2)-\mathbf{U}'_i(-z-d_1)\right]\\ &+\sum_{m=1}^{\infty}\left(\frac{\mu_1-\mu_2}{\mu_1+\mu_2}\right)^m\begin{bmatrix}\left[\mathbf{U}'_s(z-d_2-2mH)-\mathbf{U}'_s(z-d_1-2mH)\right]\\ -\left[\mathbf{U}'_i(-z-d_2+2mH)-\mathbf{U}'_i(-z-d_1+2mH)\right]\\ +\left[\mathbf{U}'_s(z-d_2+2mH)-\mathbf{U}'_s(z-d_1+2mH)\right]\\ -\left[\mathbf{U}'_i(-z-d_2-2mH)-\mathbf{U}'_i(-z-d_1-2mH)\right]\end{bmatrix}\end{aligned}\qquad(24)$$

where $\mathbf{U}'$ is the depth-integrated solution and where individual elements of the source and image tensors are

$$\mathbf{U}'_s(\mathbf{k},Z)=\begin{bmatrix}U_x & U_y & U_z\\ U_y & V_y & V_z\\ U_z & V_z & W_z\end{bmatrix}\quad\text{and}\quad\mathbf{U}'_i(\mathbf{k},Z)=\begin{bmatrix}U_x & U_y & U_z\\ U_y & V_y & V_z\\ -U_z & -V_z & -W_z\end{bmatrix},\qquad(25)$$

Here ***Z*** represents all z-dependent terms, including all combinations of z, d_n, and $2mH$. The six independent functions of the deformation tensor are

$$\begin{bmatrix} U_x \\ U_y \\ U_z \\ V_y \\ V_z \\ W_z \end{bmatrix} = \frac{C}{\beta^2} \begin{bmatrix} D + \frac{k_y^2}{|\mathbf{k}|^2} - \frac{k_x^2}{|\mathbf{k}|^2} & -\frac{k_x^2}{|\mathbf{k}|^2} \\ -2\frac{k_x k_y}{|\mathbf{k}|^2} & -\frac{k_x k_y}{|\mathbf{k}|^2} \\ -i\frac{k_x}{|\mathbf{k}|} & -i\frac{k_x}{|\mathbf{k}|} \\ D + \frac{k_x^2}{|\mathbf{k}|^2} - \frac{k_y^2}{|\mathbf{k}|^2} & -\frac{k_y^2}{|\mathbf{k}|^2} \\ -i\frac{k_y}{|\mathbf{k}|} & -i\frac{k_y}{|\mathbf{k}|} \\ D+1 & 1 \end{bmatrix} \begin{bmatrix} e^{-\beta Z} \\ \beta Z e^{-\beta Z} \end{bmatrix}, \qquad (26)$$

where $C = \frac{(\lambda + \mu)}{4\mu(\lambda + 2\mu)}$ $D = \frac{\lambda + 3\mu}{\lambda + \mu}$ $|\mathbf{k}| = \left(k_x^2 + k_y^2\right)^{1/2}$ and $\beta = 2\pi|\mathbf{k}|$.

These solutions (26) are identical to those of *Smith and Sandwell* [2003] but have been simplified here for further manipulation of the exponential terms. In particular, we analytically sum the infinite series for the case of $\mu_2 = 0$ (next section) which corresponds to the end-member case of an elastic plate over a fluid half-space.

viii) ***Analytic treatment of an infinite sum***

Inspection of the general solution above shows that when the rigidity of the half space, μ_2, goes to zero, the convergence of the layered solution become problematic. We can treat the infinite sum analytically by summing the terms from the deformation tensor (26) that are dependent upon *Z:*

$$e^{-\beta Z} \quad \text{and} \quad \beta Z e^{-\beta Z}.$$

We first analytically treat the simple exponential, $e^{-\beta Z}$:

$$\sum_{m=1}^{\infty} e^{-\beta(z'+2mH)} = e^{-\beta z'} e^{-\beta 2H} \sum_{m=0}^{\infty} e^{-\beta 2mH} = e^{-\beta z'} \frac{e^{-\beta 2H}}{1 - e^{-\beta 2H}} \tag{27}$$

where *z'* now represents all terms of the form '$z - d_n$', as in those of the deformation tensor. Next we treat the $\beta Z e^{-\beta Z}$ term by noting that

$$-\beta \frac{\partial}{\partial \beta} e^{-\beta Z} = \beta Z e^{-\beta Z}. \tag{28}$$

Therefore, to evaluate the sum of $\beta Z e^{-\beta Z}$, we take the derivative of the sum of $e^{-\beta Z}$ with respect to β:

$$\sum_{m=1}^{\infty} \beta(z' + 2mH) e^{-\beta(z'+2mH)} = -\beta \frac{\partial}{\partial \beta} \sum_{m=1}^{\infty} e^{-\beta(z'+2mH)} = -\beta \frac{\partial}{\partial \beta} \frac{e^{-\beta(z'+2H)}}{1 - e^{-\beta 2H}}$$

$$= \frac{e^{-\beta(z'+2H)}}{\left(1 - e^{-\beta 2H}\right)} \left[\beta\left(z' + 2H\right) + \frac{2\beta H e^{-\beta 2H}}{\left(1 - e^{-\beta 2H}\right)} \right]. \tag{29}$$

For the special case of an elastic plate over a fluid half-space ($\mu_2 = 0$), computing the infinite sum of the general solution (24-26) is not necessary, as the reduced equations above (28-29) can be used instead to increase the computational convergence speed.

ix) ***2-D analytic test for a fault-plane***

We will now compare our Fourier-derived layered half-space solution for a fault plane with the analytic solution for a layered half-space.

The analytic solution from *Rybicki* is:

$$V = \frac{V_o}{\pi}\left[\tan^{-1}\left(\frac{d_2}{x}\right) - \tan^{-1}\left(\frac{d_1}{x}\right) + \sum_{m=1}^{\infty}\left(\frac{\mu_1 - \mu_2}{\mu_1 + \mu_2}\right)^m \left\{\begin{array}{l}\tan^{-1}\left(\frac{d_2 - 2mH}{x}\right) - \tan^{-1}\left(\frac{d_1 - 2mH}{x}\right) \\ + \tan^{-1}\left(\frac{d_2 + 2mH}{x}\right) - \tan^{-1}\left(\frac{d_1 + 2mH}{x}\right)\end{array}\right\}\right] \tag{30}$$

where V_o= slip

d_1 = lower locking depth

d_2 = upper locking depth

H = boundary between rigidities μ_1 and μ_2.

For the following combinations of lame parameters, for a fault locked between $d_1 = 10$ and $d_2 = 1$, layer $H = 30$, the analytic fault solution behaves according to Figure (7):

$\lambda = \mu_1 = \mu_2$	homogeneous half - space
$\lambda = \mu_1 = 1, \quad \mu_2 = 0$	elastic plate over fluid half - space
$\lambda = \mu_1 = 1, \quad \mu_2 = 3.25$	layered half - space

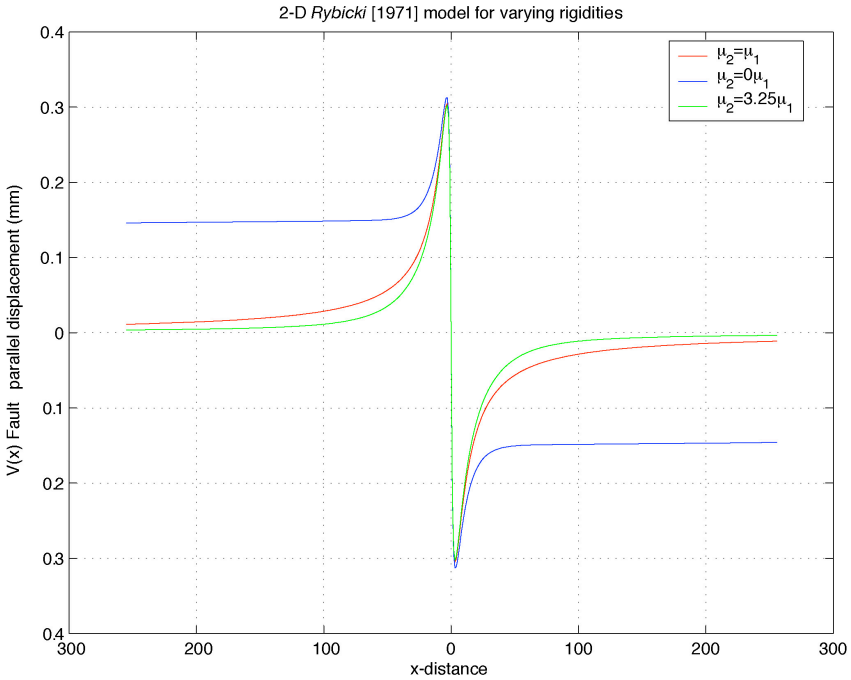


Figure 7. *Rybicki* [1971] model for varying rigidities

We again compare *Rybicki's* 2-D analytic function (30) to our 3-D Fourier solution for a layered half-space, now for an infinitely-long fault-plane (24-26). In order to form a double-couple dislocation, we have again constructed a screw dislocation by taking the derivative of the point source in the direction normal to the fault plane. The fault plane is imbedded in a 2-D grid which is Fourier transformed, multiplied by the transfer functions above, and inverse Fourier transformed.

For comparison with the analytic solution of *Rybicki* (30), we must again divide our Fourier solution by Δx which corresponds to the cell-size in the x-direction. As a first test, we set $\lambda = \mu_1 = \mu_2$ and test the case for a layered half-space of the same rigidity, better described as a homogeneous half-space. This numerical test will also be compared to both analytic solutions for a homogeneous half-space and for a layered half-space. The comparison between the three solutions is shown in Figure 8.

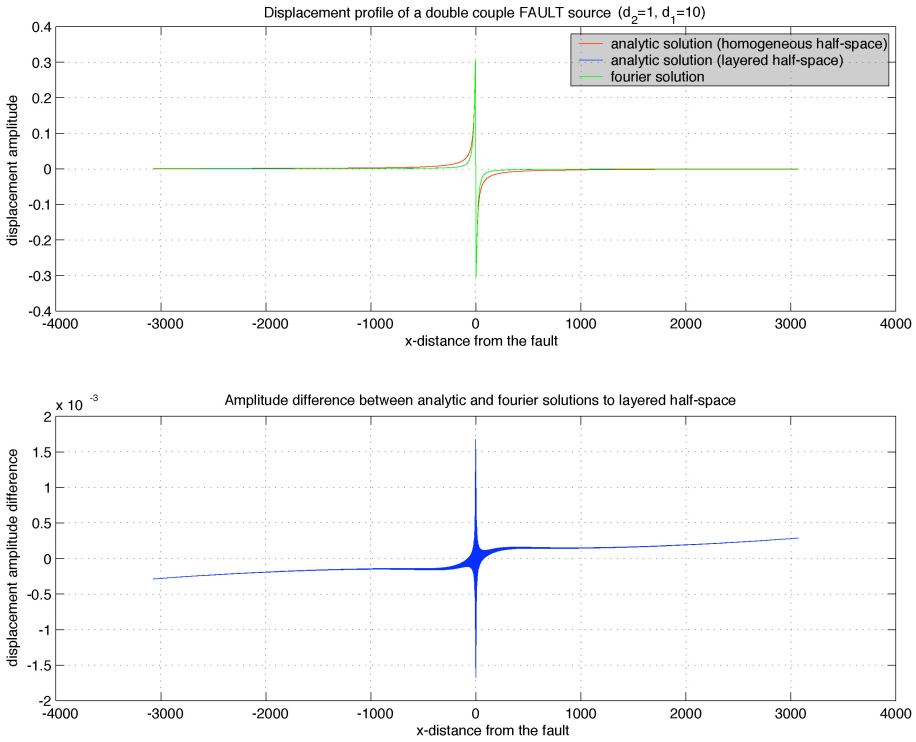


Figure 8. *Rybicki* [1971] and Fourier models for half-space simulation

For a fault locked between 1-10 km, the x-length of the Fourier grid must be greater than 6144 to achieve better than 10^{-3} numerical accuracy. A smaller dimension can be used if the fault has a finite length in the y-direction.

As a second test, we will inspect the case for a layered half-space with the half-space rigidity set to zero, simulating an elastic plate. We set $\lambda = \mu_1 = 1$, $\mu_2 = 0$ and and plot the results in Figure 9.

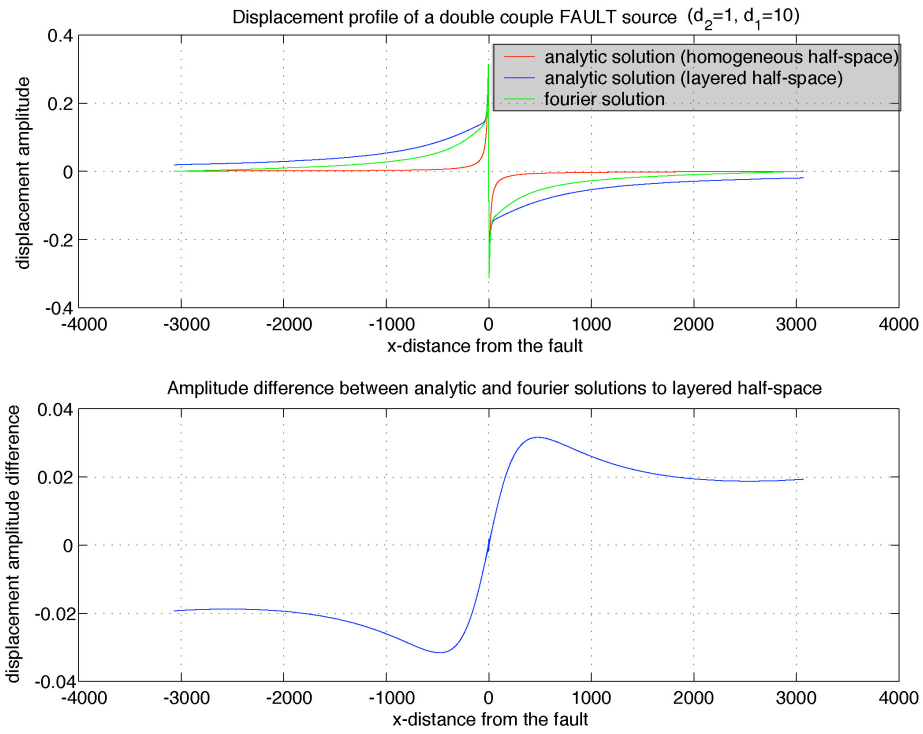


Figure 9. *Rybicki* [1971] and Fourier models for plate simulations (*m*=5)

From Figure 9, we again note that the analytical solution and Fourier solutions are dependent on *m* terms summed. Figure 10 shows how these solutions converge as $m \to \infty$

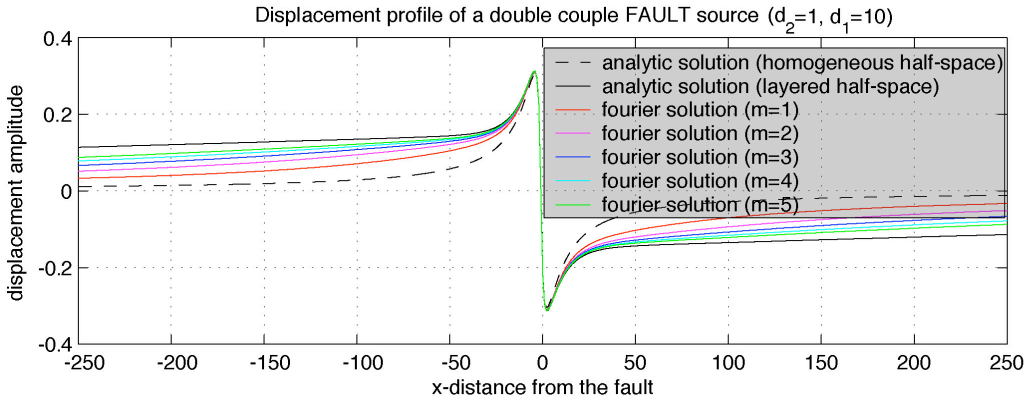


Figure 10. Convergence of *m* terms for Fourier solution

For the case of an elastic plate, $(\mu_1 - \mu_2 / \mu_1 + \mu_2)^m$ fluctuates between +/-1, and thus the solution depends on the inner +/- $2mH$ terms to converge. Again, while the larger m terms do allow the Fourier solution to approach the analytic layered solution *near the fault*, the Fourier solution approaches zero at lengths very far from the fault, causing a residual between the two solutions.

As a third test, we will inspect the case for a layered half-space of varying rigidities. We set $\lambda = \mu_1 = 1,\ \mu_2 = 3.25$ and plot the results in Figure 11.

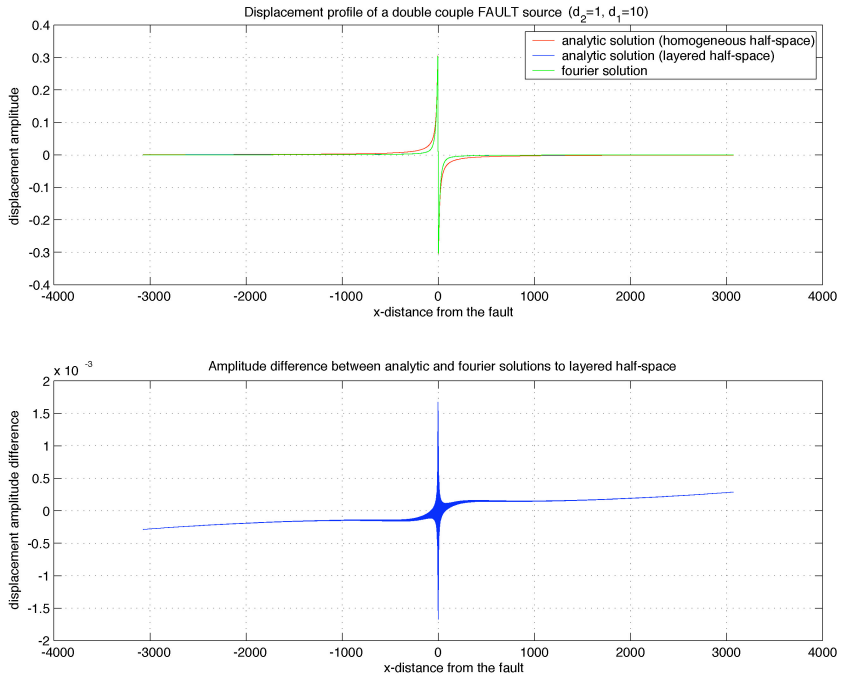


Figure 11. *Rybicki* [1971] and Fourier solution for μ_2=3.25μ_1.

For a fault locked between 1-10 km, the x-length of the Fourier grid must be greater than 6144 to achieve better than 10^{-3} numerical accuracy. A smaller dimension can be used if the fault has a finite length in the y-direction. In addition, Figure 11 shows our Fourier solution for $m = 5$ summed terms. Figure 12 shows how these solutions converge as $m \to \infty$.

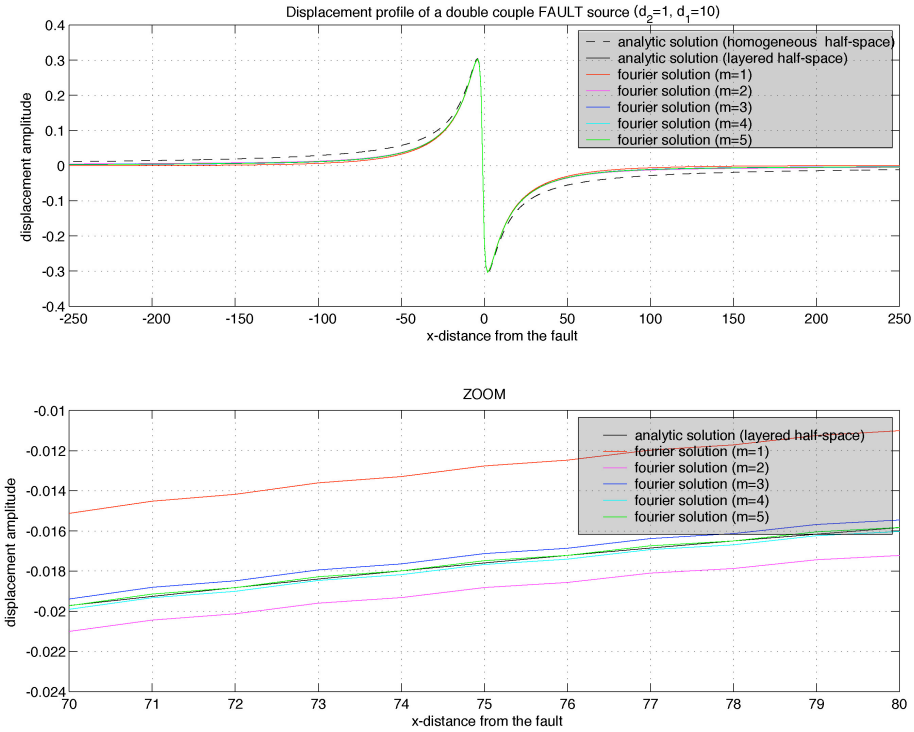


Figure 12. Convergence of *m* terms for Fourier solution

For the layered medium, $\mu_1 = 1,\ \mu_2 = 3.25$, $(\mu_1 - \mu_2 / \mu_1 + \mu_2)^m$ converges quickly, as demonstrated by Figure 12. The coefficient converges to zero for terms $> m = 10$.

x) ***Determine the surface stress distribution of a layered elastic half-space to satisfy boundary conditions - The Boussinesq Problem revisited***

As was true for the homogeneous half-space problem, the free-surface boundary condition is resolved by solving for the response of a negative surface traction, this time applied to an elastic layer overlying a viscoelastic half-space, assuming a single Maxwell relaxation time. We have included multiple image sources to *partially* satisfy this boundary condition, but again, the normal tractions remain and must be removed. We will now solve the Boussinesq Problem for a layered elastic half-space, which will require an almost identical derivation to that of the previous half-space solution, except that different boundary conditions will apply.

We will again follow the approach of *Steketee* [1958], but also include the boundary conditions discussed by *Burmister, D.M, The Theory of Stress and Displacements in Layered Systems and Applications to the Design of Airport Runways, Proceedings of the annual meeting – Highway Research Board, V 23, 1943*. Our approach differs from *Burmister's* in that we have: (1) derived the solution in Cartesian coordinates and (2) allowed all Lame parameters μ_1, μ_2, λ_1, and λ_2 to remain non-zero and non-infinite (ie, not requiring Poisson's ratio to equal 0.5).

As described by *Burmister*, the solution of the problem must satisfy the following boundary conditions:

1. The surface layer must be free shearing stresses and normal stresses, except those imposed to balance the remaining normal tractions (τ_{33}) and provide gravitational support, if necessary.

$$\tau_{zz1} = -\tau_{33} + \rho g W_1\Big|_{z=0} \qquad \tau_{xz1} = \tau_{yz1} = 0\Big|_{z=0}$$

2. Stress and displacement across the layer interface must be continuous.

$$\begin{aligned} \tau_{xz1} &= \tau_{xz2}\Big|_{z=-h} & U_1 &= U_2\Big|_{z=-h} \\ \tau_{yz1} &= \tau_{yz2}\Big|_{z=-h} & V_1 &= V_2\Big|_{z=-h} \\ \tau_{zz1} &= \tau_{zz2}\Big|_{z=-h} & W_1 &= W_2\Big|_{z=-h} \end{aligned} \tag{31}$$

3. At infinite depth, the stresses and displacements of the layer must go to zero.

$$\tau_{xz2} = \tau_{yz2} = \tau_{zz2} = 0\Big|_{z=-\infty} \qquad U_2 = V_2 = W_2 = 0\Big|_{z=-\infty}$$

Our approach is to again find the Galerkin vector $\mathbf{\Gamma}_i$, so that we may define complimentary solutions satisfying the above boundary conditions for both displacements and stresses in both the layer (layer 1) and the half-space below (layer 2). We begin by writing both displacement and stress in terms of the Galerkin vector:

$$u_i = \Gamma_{l,kk} - \alpha\Gamma_{k,ki}$$
$$\tau_{ij} = \lambda(1-\alpha)\delta_{ij}\Gamma_{k,kll} + \mu\left(\Gamma_{i,ikk} + \Gamma_{j,jkk}\right) - 2\mu\alpha\Gamma_{k,kij} \tag{32}$$

or more explicitly,

$$U_B = -\alpha\frac{\partial^2\Gamma}{\partial x\partial z} \qquad V_B = -\alpha\frac{\partial^2\Gamma}{\partial y\partial z} \qquad W_B = (1-\alpha)\frac{\partial^2\Gamma}{\partial z^2} - \beta^2\Gamma \tag{33}$$
$$\tau_{xz} = \mu\frac{\partial}{\partial x}\left\{\nabla^2\Gamma - 2\alpha\frac{\partial^2\Gamma}{\partial z^2}\right\} \qquad \tau_{yz} = \mu\frac{\partial}{\partial y}\left\{\nabla^2\Gamma - 2\alpha\frac{\partial^2\Gamma}{\partial z^2}\right\} \qquad \tau_{zz} = \mu\frac{\partial}{\partial x}\left\{\left(\frac{\alpha}{\eta}\right)\nabla^2\Gamma - 2\alpha\frac{\partial^2\Gamma}{\partial z^2}\right\}$$

where $\alpha = \dfrac{(\lambda+\mu)}{(\lambda+2\mu)}$ $\eta = \dfrac{(\lambda+\mu)}{(3\lambda+4\mu)}$ and $\beta = 2\pi|\mathbf{k}|$.

As described by *Love* [1923] and *Timoshenko* [1934], the stress and displacement equations of elasticity must satisfy the equation of compatibility, most commonly known as the biharmonic equation:

$$\nabla^4\Gamma = \frac{\partial}{\partial x^2}\left(\nabla^2\Gamma\right) + \frac{\partial}{\partial y^2}\left(\nabla^2\Gamma\right) + \frac{\partial}{\partial z^2}\left(\nabla^2\Gamma\right) = 0. \tag{34}$$

The general solution to this problem is

$$\Gamma = (A + Cz)e^{\beta z} - (B + Dz)e^{-\beta z}. \tag{35}$$

For the layered case, layers 1 and 2 will have the following representation:

$$\Gamma_1 = (A_1 + C_1 z)e^{\beta z} - (B_1 + D_1 z)e^{-\beta z} \qquad \Gamma_2 = (A_2 + C_2 z)e^{\beta z} - (B_2 + D_2 z)e^{-\beta z}. \tag{36}$$

Likewise, the two layers will have displacements and stresses as functions of the respective Galerkin vectors:

$$\begin{aligned} &\text{layer 1}: \; U_{B1}(\Gamma_1),\; V_{B1}(\Gamma_1),\; W_{B1}(\Gamma_1),\; \tau_{xz1}(\Gamma_1),\; \tau_{yz1}(\Gamma_1),\; \tau_{zz1}(\Gamma_1) \\ &\text{layer 2}: \; U_{B2}(\Gamma_2),\; V_{B2}(\Gamma_2),\; W_{B2}(\Gamma_2),\; \tau_{xz2}(\Gamma_2),\; \tau_{yz2}(\Gamma_2),\; \tau_{zz2}(\Gamma_2) \end{aligned}.$$

In order to ensure that our solution is valid for $z < 0$, we need to make coeffiecients B_2 and $D_2 = 0$ in order to suppress the two solutions that diverge as $z \to -\infty$. For $B_2 = D_2 = 0$, we have

$$\Gamma_1 = (A_1 + C_1 z)e^{\beta z} - (B_1 + D_1 z)e^{-\beta z} \qquad \text{and} \qquad \Gamma_2 = (A_2 + C_2 z)e^{\beta z}. \tag{37}$$

By substituting the above Galerkin vectors and their associated derivatives into the equations for stresses and displacements in both layers, we have

$$
\begin{aligned}
U_{B1} &= -i2\pi k_x \alpha_1 \left[A_1 \beta e^{\beta z} + B_1 \beta e^{-\beta z} + C_1 (1+\beta z) e^{\beta z} - D_1 (1-\beta z) e^{-\beta z} \right] \\
V_{B1} &= -i2\pi k_y \alpha_1 \left[A_1 \beta e^{\beta z} + B_1 \beta e^{-\beta z} + C_1 (1+\beta z) e^{\beta z} - D_1 (1-\beta z) e^{-\beta z} \right] \\
W_{B1} &= -\beta \alpha_1 \left[A_1 \beta e^{\beta z} - B_1 \beta e^{-\beta z} + C_1 (2+\beta z - 2/\alpha_1) e^{\beta z} + D_1 (2-\beta z - 2/\alpha_1) e^{-\beta z} \right]
\end{aligned}
$$

$$
\begin{aligned}
U_{B2} &= -i2\pi k_x \alpha_2 \left[A_2 \beta e^{\beta z} + C_2 (1+\beta z) e^{\beta z} \right] \\
V_{B2} &= -i2\pi k_y \alpha_2 \left[A_2 \beta e^{\beta z} + C_2 (1+\beta z) e^{\beta z} \right] \\
W_{B2} &= -\beta \alpha_2 \left[A_2 \beta e^{\beta z} + C_2 (2+\beta z - 2/\alpha_2) e^{\beta z} \right]
\end{aligned}
$$

$$
\begin{aligned}
\tau_{xz1} &= -i4\pi k_x \mu_1 \alpha_1 \beta \left[A_1 \beta e^{\beta z} - B_1 \beta e^{-\beta z} + C_1 (2+\beta z - 1/\alpha_1) e^{\beta z} + D_1 (2-\beta z - 1/\alpha_1) e^{-\beta z} \right] \\
\tau_{yz1} &= -i4\pi k_y \mu_1 \alpha_1 \beta \left[A_1 \beta e^{\beta z} - B_1 \beta e^{-\beta z} + C_1 (2+\beta z - 1/\alpha_1) e^{\beta z} + D_1 (2-\beta z - 1/\alpha_1) e^{-\beta z} \right] \\
\tau_{zz1} &= -2\mu_1 \alpha_1 \beta^2 \left[A_1 \beta e^{\beta z} + B_1 \beta e^{-\beta z} + C_1 (3+\beta z - 1/\eta_1) e^{\beta z} - D_1 (3-\beta z - 1/\eta_1) e^{-\beta z} \right]
\end{aligned}
$$

$$
\begin{aligned}
\tau_{xz2} &= -i4\pi k_x \mu_2 \alpha_2 \beta \left[A_2 \beta e^{\beta z} + C_2 (2+\beta z - 1/\alpha_2) e^{\beta z} \right] \\
\tau_{yz2} &= -i4\pi k_y \mu_2 \alpha_2 \beta \left[A_2 \beta e^{\beta z} + C_2 (2+\beta z - 1/\alpha_2) e^{\beta z} \right] \\
\tau_{zz2} &= -2\mu_2 \alpha_2 \beta^2 \left[A_2 \beta e^{\beta z} + C_2 (3+\beta z - 1/\eta_2) e^{\beta z} \right]
\end{aligned}
\tag{38}
$$

We will now use these solutions and the appropriate boundary conditions to solve for the six coefficients of A_1, B_1, C_1, D_1, A_2, C_2. Noting the symmetry between the U_B, V_B and τ_{xz}, τ_{yz} components, we can reduce our set of boundary conditions from nine to six:

$$
\begin{aligned}
&\tau_{zz1} = -\tau_{33} + \rho g W_1 \big|_{z=0} \qquad && \tau_{xz1} = 0 \big|_{z=0} \\
&\tau_{xz1} = \tau_{xz2} \big|_{z=-h} && \tau_{zz1} = \tau_{zz2} \big|_{z=-h} \\
&U_1 = U_2 \big|_{z=-h} && W_1 = W_2 \big|_{z=-h}
\end{aligned}
\quad . \tag{39}
$$

We now have 6 equations and 6 unknown coefficients. We will solve for each unknown by solving the least-squares problem. We will now make the appropriate substitutions for each boundary condition:

(1) For $\tau_{zz1} = -\tau_{33} + \rho g W_1 \big|_{z=0}$ (40)

$$
1 = \left. \frac{-\tau_{zz1} + \rho g W_1}{\tau_{33}} \right|_{z=0}
$$

$$
1 = \begin{bmatrix} A_1 \chi \beta (2\mu_1 \beta - \rho g) + B_1 \chi \beta (2\mu_1 \beta + \rho g) + 2C_1 \chi \left(\mu_1 \beta (3 - 1/\eta_1) - \rho g (1 - 1/\alpha_1) \right) \\ -2D_1 \chi \left(\mu_1 \beta (3 - 1/\eta_1) + \rho g (1 - 1/\alpha_1) \right) \end{bmatrix}
$$

where $\chi = \dfrac{\beta \alpha_1}{\tau_{33}}$.

Note that this boundary condition will only be used for solutions requiring a gravity force balance. Those solutions without gravity will have ρg set to zero.

(2) For $\tau_{xz1} = 0\big|_{z=0}$ (41)

$$0 = -i4\pi k_x \mu_1 \alpha_1 \beta \left[A_1 \beta - B_1 \beta + C_1 \left(2 - 1/\alpha_1\right) + D_1 \left(2 - 1/\alpha_1\right) \right].$$

(3) For $\tau_{xz1} = \tau_{xz2}\big|_{z=-h}$ (42)

$$0 = -i4\pi k_x \beta \begin{bmatrix} A_1 \mu_1 \alpha_1 \beta e^{-\beta h} - B_1 \mu_1 \alpha_1 \beta e^{\beta h} + C_1 \mu_1 \alpha_1 \left(2 - \beta h - 1/\alpha_1\right) e^{-\beta h} + D_1 \mu_1 \alpha_1 \left(2 + \beta h - 1/\alpha_1\right) e^{\beta h} \\ -A_2 \mu_2 \alpha_2 \beta e^{-\beta h} - C_2 \mu_2 \alpha_2 \left(2 - \beta h - 1/\alpha_2\right) e^{-\beta h} \end{bmatrix}$$

(4) For $\tau_{zz1} = \tau_{zz2}\big|_{z=-h}$ (43)

$$0 = -2\beta^2 \begin{bmatrix} A_1 \mu_1 \alpha_1 \beta e^{-\beta h} + B_1 \mu_1 \alpha_1 \beta e^{\beta h} + C_1 \mu_1 \alpha_1 \left(3 - \beta h - 1/\eta_1\right) e^{-\beta h} - D_1 \mu_1 \alpha_1 \left(3 + \beta h - 1/\eta_1\right) e^{\beta h} \\ -A_2 \mu_2 \alpha_2 \beta e^{-\beta h} - C_2 \mu_2 \alpha_2 \left(3 - \beta h - 1/\eta_2\right) e^{-\beta h} \end{bmatrix}$$

(5) For $U_1 = U_2\big|_{z=-h}$ (44)

$$0 = -i2\pi k_x \begin{bmatrix} A_1 \alpha_1 \beta e^{-\beta h} + B_1 \alpha_1 \beta e^{\beta h} + C_1 \alpha_1 \left(1 - \beta h\right) e^{-\beta h} - D_1 \alpha_1 \left(1 + \beta h\right) e^{\beta h} \\ -A_2 \alpha_2 \beta e^{-\beta h} - C_2 \alpha_2 \left(1 - \beta h\right) e^{-\beta h} \end{bmatrix}$$

(6) For $W_1 = W_2\big|_{z=-h}$ (45)

$$0 = -\beta \begin{bmatrix} A_1 \alpha_1 \beta e^{-\beta h} - B_1 \alpha_1 \beta e^{\beta h} + C_1 \alpha_1 \left(2 - \beta h - 2/\alpha_1\right) e^{-\beta h} + D_1 \alpha_1 \left(2 + \beta h - 2/\alpha_1\right) e^{\beta h} \\ -A_2 \alpha_2 \beta e^{-\beta h} - C_2 \alpha_2 \left(2 - \beta h - 2/\alpha_2\right) e^{-\beta h} \end{bmatrix}$$

Plugging in these equations, our least squares matrix then looks like:

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \chi\beta\left(2\mu_1\beta - \rho g\right) & \chi\beta\left(2\mu_1\beta + \rho g\right) & 2\chi\left(\mu_1\beta\left(3 - 1/\eta_1\right) - \rho g\left(1 - 1/\alpha_1\right)\right) & -2\chi\left(\mu_1\beta\left(3 - 1/\eta_1\right) + \rho g\left(1 - 1/\alpha_1\right)\right) & 0 & 0 \\ \beta & -\beta & \left(2 - 1/\alpha_1\right) & \left(2 - 1/\alpha_1\right) & 0 & 0 \\ \mu_1\alpha_1\beta e^{-\beta h} & -\mu_1\alpha_1\beta e^{\beta h} & \mu_1\alpha_1\left(2 - \beta h - 1/\alpha_1\right)e^{-\beta h} & \mu_1\alpha_1\left(2 + \beta h - 1/\alpha_1\right)e^{\beta h} & -\mu_2\alpha_2\beta e^{-\beta h} & -\mu_2\alpha_2\left(2 - \beta h - 1/\alpha_2\right)e^{-\beta h} \\ \mu_1\alpha_1\beta e^{-\beta h} & \mu_1\alpha_1\beta e^{\beta h} & \mu_1\alpha_1\left(3 - \beta h - 1/\eta_1\right)e^{-\beta h} & -\mu_1\alpha_1\left(3 + \beta h - 1/\eta_1\right)e^{\beta h} & -\mu_2\alpha_2\beta e^{-\beta h} & -\mu_2\alpha_2\left(3 - \beta h - 1/\eta_2\right)e^{-\beta h} \\ \alpha_1\beta e^{-\beta h} & \alpha_1\beta e^{\beta h} & \alpha_1\left(1 - \beta h\right)e^{-\beta h} & -\alpha_1\left(1 + \beta h\right)e^{\beta h} & -\alpha_2\beta e^{-\beta h} & -\alpha_2\left(1 - \beta h\right)e^{-\beta h} \\ \alpha_1\beta e^{-\beta h} & -\alpha_1\beta e^{\beta h} & \alpha_1\left(2 - \beta h - 2/\alpha_1\right)e^{-\beta h} & \alpha_1\left(2 + \beta h - 2/\alpha_1\right)e^{\beta h} & -\alpha_2\beta e^{-\beta h} & -\alpha_2\left(2 - \beta h - 2/\alpha_2\right)e^{-\beta h} \end{bmatrix} \begin{bmatrix} A_1 \\ B_1 \\ C_1 \\ D_1 \\ A_2 \\ C_2 \end{bmatrix}$$

(46)

Solving for coefficients $A_1,\ B_1,\ C_1,\ D_1,\ A_2,\ C_2,$ we explore four cases:

(1) General case with $\lambda_1,\ \lambda_2,\ \mu_1,\ \mu_2$ (no gravity)
(2) General case with $\lambda_1,\ \lambda_2,\ \mu_1,\ \mu_2$ *and* gravity force balance
(3) Check of elastic half-space solution: $\lambda_1 = \lambda_2$ and $\mu_1 = \mu_2$ (no gravity)
(4) Special case for an elastic plate: $\mu_2 = 0$ *and* gravity force balance

Case (1) $\lambda_1,\ \lambda_2,\ \mu_1,\ \mu_2$ (no gravity)

(47)

$$A_1 = \frac{1}{d}\beta^2 \frac{(\lambda_2+\mu_2)}{(\lambda_1+2\mu_1)^2(\lambda_2+2\mu_2)^2}\left\{\begin{array}{l}
\mu_1^2\lambda_2(\lambda_1+\mu_1)\left[2\mu_1\left(\beta^2h^2e^{-2\beta h}\right)-\lambda_1\left(1-e^{-2\beta h}\left(1-2\beta h+2\beta^2h^2\right)\right)\right] \\
-\mu_2^2\lambda_1(\lambda_2+\mu_2)\left[\lambda_1\left(1+e^{-2\beta h}\left(1-2\beta h+2\beta^2h^2\right)\right)\right] \\
-\mu_1\mu_2\left\{\begin{array}{l}
\mu_1\left[\begin{array}{l} 4\lambda_1\lambda_2-8\mu_1\mu_2e^{-2\beta h}\left(1-\beta^2h^2\right) \\ +3(\lambda_1+\mu_1)\left\{\begin{array}{l}\lambda_1\left(1-e^{-2\beta h}\left(1-2\beta h+2\beta^2h^2\right)\right)\\ -2\mu_1\beta^2h^2e^{-2\beta h}\end{array}\right.\end{array}\right] \\
+\mu_2\left[\begin{array}{l} 4\lambda_1^2\left(1-e^{-2\beta h}\left(\beta h+\beta^2h^2\right)\right) \\ +\lambda_1(\lambda_2+\mu_2)\left(3-e^{-2\beta h}\left(3-2\beta h+4\beta^2h^2\right)\right)\end{array}\right] \\
+2\mu_1\mu_2\left[\begin{array}{l}\lambda_1\left(5-2e^{-2\beta h}\left(1+\beta h+2\beta^2h^2\right)\right)\\ +2\mu_1e^{-2\beta h}\left(1-\beta^2h^2\right)\\ +(\lambda_2+\mu_2)e^{-2\beta h}\left(2+\beta^2h^2\right)\end{array}\right] \\
+2\lambda_1^2\lambda_2
\end{array}\right.
\end{array}\right\}$$

$$B_1 = \frac{1}{d}e^{-2\beta h}\beta^2 \frac{(\lambda_2+\mu_2)}{(\lambda_1+2\mu_1)^2(\lambda_2+2\mu_2)^2}\left\{\begin{array}{l}
\mu_1^2\lambda_2(\lambda_1+\mu_1)\left[\begin{array}{l}\lambda_1\left(1+2\beta h(1+\beta h)-e^{-2\beta h}\right)\\ +2\mu_1\left(\beta^2h^2\right)\end{array}\right] \\
-\mu_2^2\lambda_1(\lambda_2+\mu_2)\left[\lambda_1\left(1+2\beta h(1+2\beta h)+e^{-2\beta h}\right)\right] \\
+\mu_1\mu_2\left\{\begin{array}{l}
\mu_1\left[\begin{array}{l} 6\mu_1^2\beta^2h^2+4\lambda_1\lambda_2e^{-2\beta h} \\ +3\mu_1\lambda_1\left(1+2\beta h(1+2\beta h)-e^{-2\beta h}\right) \\ +3\lambda_1^2\left(1+2\beta h(1+\beta h)-e^{-2\beta h}\right)\end{array}\right] \\
-\mu_2\left[\begin{array}{l} 4\lambda_1^2\left(\beta h(1+\beta h)-e^{-2\beta h}\right) \\ +\lambda_1(\lambda_2+\mu_2)\left(3+2\beta h(1+2\beta h)+3e^{-2\beta h}\right)\end{array}\right] \\
+2\mu_1\mu_2\left[\begin{array}{l}\lambda_1\left(2-2\beta h(1+2\beta h)+3e^{-2\beta h}\right)\\ +2\mu_1\left(1-\beta^2h^2\right)\\ -(\lambda_2+\mu_2)\left(2+\beta^2h^2\right)\end{array}\right] \\
+2\lambda_1^2\lambda_2e^{-2\beta h}
\end{array}\right.
\end{array}\right\}$$

$$C_1 = \frac{1}{d}\beta^3 \frac{(\lambda_1+\mu_1)(\lambda_2+\mu_2)}{(\lambda_1+2\mu_1)^2(\lambda_2+2\mu_2)^2}\left\{\begin{array}{l}\mu_1^2\lambda_2(\lambda_1+\mu_1)\left[1-e^{-2\beta h}(1-2\beta h)\right]\\ +\mu_2^2\lambda_1(\lambda_2+\mu_2)\left[1+e^{-2\beta h}(1-2\beta h)\right]\\ +\mu_1\mu_2\left\{\begin{array}{l}\mu_1\left[3(\lambda_1+\mu_1)\left\{1-e^{-2\beta h}(1-2\beta h)\right\}\right]\\ +\mu_2\left[\begin{array}{l}(\lambda_2+\mu_2)\left(3+e^{-2\beta h}(1-2\beta h)\right)\\ +2\lambda_1\left(2+e^{-2\beta h}(1-2\beta h)\right)\end{array}\right]\\ +2\mu_1\mu_2\left[5+e^{-2\beta h}(1-2\beta h)\right]\\ +2\lambda_2(\lambda_1+2\mu_1)\end{array}\right.\end{array}\right.$$

$$D_1 = \frac{1}{d}e^{-2\beta h}\beta^3 \frac{(\lambda_1+\mu_1)(\lambda_2+\mu_2)}{(\lambda_1+2\mu_1)^2(\lambda_2+2\mu_2)^2}\left\{\begin{array}{l}\mu_1^2\lambda_2(\lambda_1+\mu_1)\left[1+2\beta h-e^{-2\beta h}\right]\\ -\mu_2^2\lambda_1(\lambda_2+\mu_2)\left[1+2\beta h+e^{-2\beta h}\right]\\ +\mu_1\mu_2\left\{\begin{array}{l}\mu_1\left[3(\lambda_1+\mu_1)\left\{1+2\beta h-e^{-2\beta h}\right\}\right]\\ -\mu_2\left[\begin{array}{l}(\lambda_2+\mu_2)\left(1+2\beta h-3e^{-2\beta h}\right)\\ +2\lambda_1\left(1+2\beta h-2e^{-2\beta h}\right)\end{array}\right]\\ -2\mu_1\mu_2\left[1+2\beta h-3e^{-2\beta h}\right]\\ +2\lambda_2(\lambda_1+2\mu_1)e^{-2\beta h}\end{array}\right.\end{array}\right.$$

$$d = \beta^4 \frac{(\lambda_1+\mu_1)(\lambda_2+\mu_2)}{(\lambda_1+2\mu_1)^2(\lambda_2+2\mu_2)^2}\frac{1}{\tau_{33}}\left\{\frac{-2\mu_1\beta}{(\lambda_1+2\mu_1)}d_1\right\}$$

where $$d_1 = \left\{\begin{array}{l}\mu_1^2(\lambda_1+\mu_1)^2(\lambda_2+3\mu_2)\left[e^{-4\beta h}-2e^{-2\beta h}\left(1+2\beta^2h^2\right)+1\right]\\ +\mu_2^2\lambda_1^2(\lambda_2+\mu_2+4\mu_1)\left[e^{-4\beta h}+2e^{-2\beta h}\left(1+2\beta^2h^2\right)+1\right]\\ +\mu_1\mu_2\left\{\begin{array}{l}\mu_1\mu_2\left\{\begin{array}{l}(\lambda_2+\mu_2)\left[3e^{-4\beta h}+2e^{-2\beta h}\left(5+2\beta^2h^2\right)+3\right]\\ -2\mu_1\left[3e^{-4\beta h}+2e^{-2\beta h}\left(1-2\beta^2h^2\right)-5\right]\\ -2\lambda_1\left[5e^{-4\beta h}+2e^{-2\beta h}\left(1-4\beta^2h^2\right)-7\right]\end{array}\right.\\ +4\mu_2\left[\begin{array}{l}\lambda_1(\lambda_2+\mu_2)\left[e^{-4\beta h}+2e^{-2\beta h}\left(1+\beta^2h^2\right)+1\right]\\ -2\lambda_1^2\left[e^{-4\beta h}+e^{-2\beta h}\left(1+\beta^2h^2\right)\right]\end{array}\right.\\ -2\lambda_2(\lambda_1+\mu_1)(\lambda_1+2\mu_1)\left(e^{-4\beta h}-1\right)\end{array}\right.\end{array}\right.$$

Case (2) $\lambda_1, \lambda_2, \mu_1, \mu_2$ *and* gravity force balance (48)

For this solution, coeffiecients $A_1, B_1, C_1, D_1, A_2, C_2$, are identical to those of Case (1) except for the "d" term, which now has an extra piece associated with the gravity contribution:

$$d = \beta^4 \frac{(\lambda_1+\mu_1)(\lambda_2+\mu_2)}{(\lambda_1+2\mu_1)^2(\lambda_2+2\mu_2)^2}\frac{1}{\tau_{33}}\left\{\frac{-2\mu_1\beta}{(\lambda_1+2\mu_1)}d_1 + \rho g d_2\right\}$$

where d_1 is the same from Case (1) and

and
$$d_2 = \left\{\begin{array}{l} -\mu_1^2\lambda_2(\lambda_1+\mu_1)\left[e^{-4\beta h} - 4\beta h e^{-2\beta h} - 1\right] \\ -\mu_2^2\lambda_1(\lambda_2+\mu_2)\left[e^{-4\beta h} + 4\beta h e^{-2\beta h} - 1\right] \\ +\mu_1\mu_2\left\{\begin{array}{l} \mu_1\left[\begin{array}{l} 4\lambda_2\left[e^{-4\beta h} + e^{4\beta h}\right] \\ -3(\lambda_1+\mu_1)\left[e^{-4\beta h} - 4\beta h e^{-2\beta h} - 1\right]\end{array}\right] \\ +\mu_2\left[\begin{array}{l} 4\lambda_1\left[e^{-4\beta h} - 2\beta h e^{-2\beta h} + 1\right] \\ -(\lambda_2+\mu_2)\left[3e^{-4\beta h} + 4\beta h e^{-2\beta h} - 3\right]\end{array}\right] \\ +2\mu_1\mu_2\left[3e^{-4\beta h} - 4\beta h e^{-2\beta h} + 5\right] \\ +2\lambda_1\lambda_2\left(e^{-4\beta h} + 1\right)\end{array}\right.\end{array}\right\}$$

Case (3) elastic half-space solution: $\lambda_1 = \lambda_2$ and $\mu_1 = \mu_2$ (no gravity) (49)

$$A_1 = \frac{1}{2\mu_1\alpha_1}\frac{\tau_{33}}{\beta^3}\left(2 - \frac{1}{\alpha_1}\right) \qquad B_1 = 0 \qquad C_1 = -\frac{1}{2\mu_1}\frac{\tau_{33}}{\beta^2}\left(\frac{1}{\alpha_1}\right) \qquad D_1 = 0$$

$$A_2 = A_1 \qquad C_2 = C_1$$

Case(4) elastic plate: $\mu_2 = 0$ *and* gravity force balance (50)

The solution is greatly simplified in the case of $\mu_2 = 0$:

$$A_1 = \frac{1}{d}\beta^2\frac{(\lambda_1+\mu_1)}{(\lambda_1+2\mu_1)^2}\left\{2\mu_1\left(\beta^2h^2e^{-2\beta h}\right) - \lambda_1\left(1 - e^{-2\beta h}\left(1 - 2\beta h + 2\beta^2h^2\right)\right)\right\}$$

$$B_1 = \frac{1}{d} e^{-2\beta h} \beta^2 \frac{(\lambda_1 + \mu_1)}{(\lambda_1 + 2\mu_1)^2} \left\{ 2\mu_1 \left(\beta^2 h^2 \right) + \lambda_1 \left(1 + 2\beta h \left(1 + 2\beta h \right) - e^{-2\beta h} \right) \right\}$$

$$C_1 = \frac{1}{d} \alpha_1^2 \beta^3 \left\{ 1 - e^{-2\beta h} \left(1 - 2\beta h \right) \right\}$$

$$D_1 = \frac{1}{d} e^{-2\beta h} \alpha_1^2 \beta^3 \left\{ 1 + 2\beta h - e^{-2\beta h} \right\}$$

$$d = -\alpha_1^2 \beta^4 \frac{1}{\tau_{33}} \left\{ 2\mu_1 \alpha_1 \beta d_1 + \rho g d_2 \right\}$$

where
$$d_1 = e^{-4\beta h} - 2e^{-2\beta h} \left(1 + 2\beta^2 h^2 \right) + 1$$
$$d_2 = e^{-4\beta h} - 4\beta h e^{-2\beta h} - 1$$

xii) ***Including time dependence: the Maxwell Model***

Following the approach of *Nur and Mavko* [1974] and *Savage and Prescott* [1978], we will now describe our approach for the development of a time-dependent model for 3-D displacement and stress caused by a dislocation in an elastic layer over a visco-elastic half space. Here we develop viscoelastic coefficients that are used in conjunction with the solutions derived above to manipulate the time-dependenced of the viscoelastidc problem. Because depth and time-dependence of the model are solved analytically, the numerical calculations involve only 2-D Fourier transforms. This model will ultimately be used to demonstrate the visco-elastic response of the earth throughout the earthquake cycle.

Our theory begins with the description of the viscoelastic behavior by a Maxwell body made up of an elastic element and a viscous element, connected in series. The viscous and elastic element can be represented mathematically by stress, σ, and stress rate, $\dot{\sigma}$, respectively:

$$\begin{aligned} \sigma &= 2\mu\varepsilon && \text{elastic element} \\ \sigma &= 2\eta\dot{\varepsilon} && \text{viscous element} \end{aligned} \tag{51}$$

The elastic element describes a relationship between strain, ε, and the shear modulus, μ, while the viscous element describes a relationship between strain rate, $\dot{\varepsilon}$, and viscosity, η. Combining both of these linear elements for a Maxwell body in series, the constituative equation becomes

$$\dot{\varepsilon} = \frac{1}{\mu}\dot{\sigma} + \frac{1}{\eta}\sigma . \tag{52}$$

In addition, the Maxwell time, τ, is a parameter used in describing the behavior of visco-elastic relaxation. In our model,

$$\tau = \frac{2\eta}{\mu} . \tag{53}$$

We now use the Correspondence Principle for relating the constituative equation for a Maxwell body (above) to our previously discussed elastic solution. The Correspondence Principle is based on the notion that when Laplace transformed, both equations for stress and strain take on the same "form" for all linear rheologies. Thus when the elastic solution is known, a corresponding solution describing a different rheology can be derived.

The Laplace Transform of the above stress-strain relation is,

$$L\{\varepsilon\} = s\varepsilon(s) = \frac{s}{\mu}\sigma(s) + \frac{1}{\eta}\sigma(s) \tag{54}$$

where s is the Laplace transform variable. Solving for stress, σ, we have

$$\sigma(s) = \frac{\mu s}{s + \frac{\mu}{\eta}} \varepsilon(s) \tag{55}$$

$s \to \infty \quad \sigma = \mu\varepsilon \quad$ high frequency
$s \to 0 \quad \sigma = 0 \quad$ zero frequency

By setting

$$\mu_2(s) = \frac{\mu s}{s + \frac{\mu}{\eta}} \tag{56}$$

where $\mu = \mu_1$, we find

$$\sigma(s) = \mu_2(s)\varepsilon(s). \tag{57}$$

Provided that the rigidity is replaced by $\mu_2(s)$, the Laplace transformed viscoelastic equation (above) now takes on the same form as that of the Laplace transformed elastic constituative equation. By use of the Correspondence Principle, the viscoelastic solution can be obtained as followed:

(1) Replace the rigidity variable in the elastic solution by the Laplace transformed rigidity variable
(2) Compute the inverse transform of the layered solution
(3) Integrate the solution (impulse response function) to obtain the response to a step function used to represent an earthquake
(4) Identify a recursion formula for rapid and convenient calculation
(5) Solve for the implied μ_2 and λ_2 associated with image in the infinite layers
(6) Ensure the bulk modulus remains constant by varying λ_2 for each μ_2

We have already shown that the layered elastic solution may be described by adding the sources and images associated with the layer in an infinite series by

$$U_x^L(Z) = U_x(z-a) + U_x(-z-a) + \sum_{m-1}^{\infty} \left(\frac{\mu_1 - \mu_2}{\mu_1 + \mu_2} \right)^m \begin{bmatrix} U_x(-z+a+2mH) \\ U_x(z-a+2mH) \\ U_x(-z-a+2mH) \\ U_x(z+a+2mH) \end{bmatrix}. \tag{58}$$

We now focus on the treatment of the rigidity ratio inside the infinite series solution by setting

$$\chi = \frac{\mu_1 - \mu_2}{\mu_1 + \mu_2}. \tag{59}$$

We will now treat χ as we did μ_2 in the above example of the Correspondence Principle. The Laplace transform of χ is

$$\chi(s) = \frac{\mu - \dfrac{\mu s}{s+\mu/\eta}}{\mu + \dfrac{\mu s}{s+\mu/\eta}} = \frac{\mu s + \dfrac{\mu^2}{\eta} - \mu s}{2\mu s + \dfrac{\mu^2}{\eta}} = \frac{\dfrac{\mu}{2\eta}}{s + \dfrac{\mu}{2\eta}}. \tag{60}$$

We now let

$$a = \frac{\mu}{2\eta} = \frac{1}{\tau}, \tag{61}$$

the inverse of the Maxell time. Then

$$\chi(s) = \frac{a}{s+a}. \tag{62}$$

So we now have

$$U_x^L(Z) = U_x(z-a) + U_x(-z-a) + \sum_{m-1}^{\infty}\left(\chi^m(s)\right)\begin{bmatrix} U_x(-z+a+2mH) \\ U_x(z-a+2mH) \\ U_x(-z-a+2mH) \\ U_x(z+a+2mH) \end{bmatrix}. \tag{63}$$

Next we to take the inverse Laplace transform of $\chi^m(s)$. The following pattern exists:

$$\begin{array}{lll} m=1 & \chi^1(s) = \dfrac{a}{s+a} & \chi^1(t) = ae^{-at} \\ m=2 & \chi^2(s) = \dfrac{a^2}{(s+a)^2} & \chi^2(t) = a^2te^{-at} \\ m=3 & \chi^3(s) = \dfrac{a^3}{(s+a)^3} & \chi^3(t) = a^3t^2e^{-at} \\ m & \chi^m(s) = \dfrac{a^m}{(s+a)^m} & \chi^m(t) = \dfrac{a^m t}{(m-2)!}^{m-1} e^{-at} \end{array} \tag{64}$$

Thus the inverse Laplace transform of $\chi^m(s)$ is

$$\chi^m(t) = \frac{a^m t}{(m-1)!}^{m-1} e^{-at}, \tag{65}$$

which is the impulse response function.

Next we integrate this impulse response function to obtain the response to a step function, *H(t),* which will represent a seismic event We let coefficients $A_m(t)$ and $B_m(t)$ describe this behavior:

$$A_m(t) = \frac{a^m}{(m-1)!}\int_0^t t^{m-1}e^{-at}dt \qquad B_m = \frac{(m-1)!}{a^m}A_m. \tag{66}$$

Thus

$$B_m(t) = \int_0^t t^{m-1}e^{-at}. \tag{67}$$

We will now integrate by parts:

Let

$$u = t^{m-1} \qquad du = (m-1)t^{m-2}$$
$$dv = e^{-at}dt \quad v = \frac{-1}{a}e^{-at}$$

$$\begin{aligned}\int_0^t t^{m-1}e^{-at}dt &= \frac{-t^{m-1}}{a}e^{-at}\Big|_0^t + \frac{m-1}{a}\int_0^t t^{m-2}e^{-at}dt \\ &= \frac{-t^{m-1}}{a}e^{-at} + \frac{m-1}{a}B_{m-1}\end{aligned}. \tag{68}$$

The recursion formula thus becomes:

$$B_m = -\frac{t^{m-1}}{a}e^{-at} + \frac{m-1}{a}B_{m-1}. \tag{69}$$

$$B_1 = \int_0^t e^{-at}dt = \frac{1}{a}\left[1-e^{-at}\right] \tag{70}$$

In summary, we have the following coefficients that will be used in the infinite series, replacing the rigidity ratio

$$\left(\frac{\mu_1-\mu_2}{\mu_1+\mu_2}\right)^m$$
$$= \quad A_m \quad = \quad \frac{a^m}{(m-1)!}B_m \tag{71}$$

where

$$a = \frac{\mu}{2\eta}$$

$$B_m(t) = -\frac{t^{m-1}}{a}e^{-at} + \frac{m-1}{a}B_{m-1} \qquad \text{and} \qquad B_1(t) = \frac{1}{a}\left[1 - e^{-at}\right]. \tag{72}$$

The first three terms in this infinite series are:

m	B_m	A_m
1	$\frac{1}{a}\left[1-e^{-at}\right]$	$\left[1-e^{-at}\right]$
2	$-\frac{t}{a}e^{-at} + \frac{1}{a^2}\left[1-e^{-at}\right]$	$-ate^{-at} + \left[1-e^{-at}\right]$
3	$-\frac{t^2}{a}e^{-at} + 2\left\{-\frac{t}{a}e^{-at} + \frac{1}{a^2}\left[1-e^{-at}\right]\right\}$	$-(at)^2\frac{e^{-at}}{2} - ate^{-at} + \left[1-e^{-at}\right]$

(73)

We next replace the previous rigidity ratio by the new A_m factors elastic layered model. The difference now is that each image will have its own implied μ_2 and λ_2. Solving for the implied μ_2 and λ_2 corresponding to each A_m, we obtain

$$\left(\frac{\mu_1 - \mu_2}{\mu_1 + \mu_2}\right)^m = A_m \qquad (\mu_1 - \mu_2) = A_m^{\frac{1}{m}}(\mu_1 + \mu_2)$$

$$\mu_2 = \mu_1\left(\frac{1 - A_m^{\frac{1}{m}}}{1 + A_m^{\frac{1}{m}}}\right) \tag{74}$$

We also must ensure that the bulk modulus remains constant for all $\lambda_1, \mu_1, \lambda_2, \mu_2$.

$$\kappa = \lambda + \frac{2}{3}\mu \qquad \kappa_2 = \lambda_2 + \frac{2}{3}\mu_2 = \text{constant}$$

If we set $\mu_2 = \mu_1\left(\frac{1 - A_m^{\frac{1}{m}}}{1 + A_m^{\frac{1}{m}}}\right)$, then $\lambda_2 = \kappa_2 - \frac{2}{3}\left(\frac{1 - A_m^{\frac{1}{m}}}{1 + A_m^{\frac{1}{m}}}\right)$. (75)

Figure 13 demonstrates the dependence of A_m and the ratio of μ_2 / μ_1 on m and Maxwell time t.

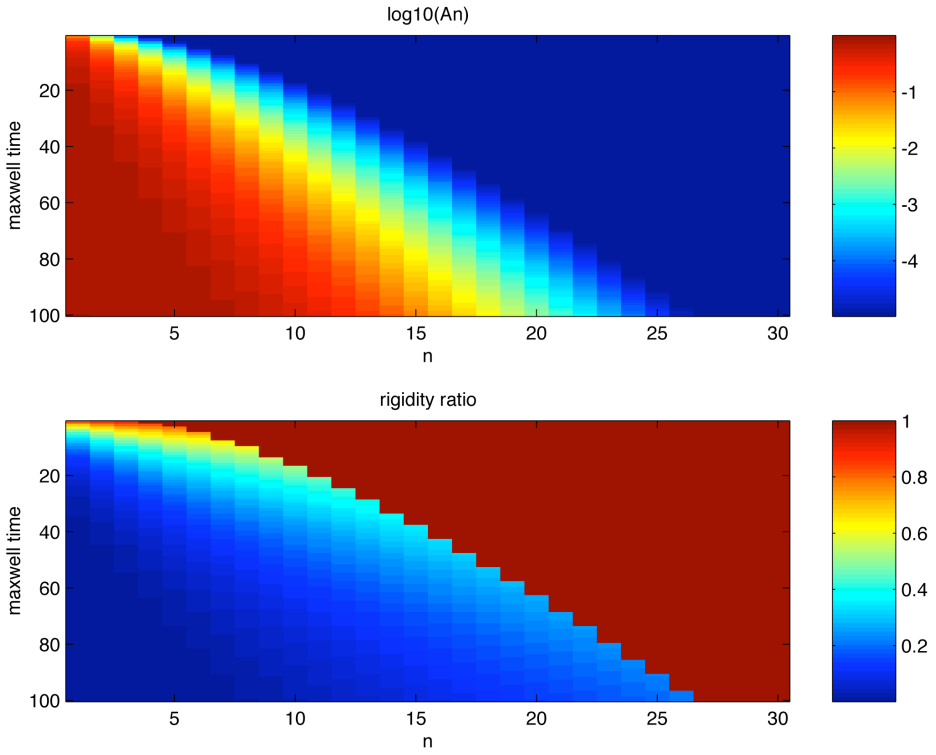


Figure 13. A_n and rigidity ratio as a function of Maxwell time and n terms.

In addition, we must also note that the vertical Boussinesq load assumes the shear response of a single Maxwell time. Since we have only solved the vertical loading problem for an elastic plate over a viscoelastic half-space, we must select the most appropriate viscosity or Maxwell time. We choose the viscosity of the uppermost layer (m = 1) noting that this is an approximation to the exact layered behavior. This approach follows the time-dependent loading problem discussed by *Brotchie and Silvester* [1969] where loading is scaled by a similar viscosity-dependent coefficient. If a vertical load is applied at $t = 0$ and the initial elastic response is described as $W(\mathbf{k},0)$, then the long-term response of the elastic plate over a fluid half-space is $W(\mathbf{k},\infty)$. For a Maxwell time of $\tau_m = 2\eta/\mu$, the viscoelastic response becomes

$$W(\mathbf{k},t) = W(\mathbf{k},0) + \left[1 - e^{-\frac{t}{\tau_m}}\right]\left[W(\mathbf{k},\infty) - W(\mathbf{k},0)\right]$$

$$= W(\mathbf{k},0)e^{-\frac{t}{\tau_m}} + \left[1 - e^{-\frac{t}{\tau_m}}\right]W(\mathbf{k},\infty). \tag{76}$$

By assuming $\mu_2(t)$ from $A_m(1)$, the Maxwell coefficients become:

$$A_m(1) = \left(1 - e^{-\frac{t}{\tau_m}}\right) \tag{77}$$

$$\mu_2 = \mu_1\left(\frac{e^{-\frac{t}{\tau_m}}}{2 - e^{-\frac{t}{\tau_m}}}\right). \tag{78}$$

As a check, we can verify that shear time variations are consistent with those expected for end-member models. For example, for times approaching zero, shear moduli of the layer and half-space are equal ($\mu_2 = \mu_1$). Alternatively, for times approaching infinity, the shear modulus of the half-space goes to zero ($\mu_2 = 0$).

xiii) ***Fortran source code for viscoelastic layered model***

The following Fortran code maxwell.f (and subroutines element.f, coefan.f, fplate.f, fvisco.f, fterm.f, boussinesql.f, coefl.c, halfspace.h, layered.h, layered_pg.h, plate.h, and coulomb.f) are used to compute displacement and stress of a viscoelastic medium from a set of input fault parameters and write these solutions to appropriate grd files. The user sets the fault dimensions, location, and grid spacing from the command line:

```
%
%  maxwell F D1 D2 Z T dble ele.dat istr U.grd V.grd W.grd H η ν f_d
%
```

```
F       - factor to multiply output files
D1, D2  - depth to bottom and top of fault (neg)
Z       - depth of observation plane
T       - time since earthquake (yr)
dble    - (0-single couple, 1-double couple)
ele.dat - file of: x1,x2,y1,y2,F1,F2,F3
istress - (0)-disp. U,V,W
          (1)-stress Txx,Tyy,Txy
          (2)-stress Tnormal,Tshear,Tcoulomb
x,y,z.grd - output files of disp. or stress
H        - plate thickness (default H = 50 km)
η        - half-space viscosity (default η = 10^19)
ν        - poisson's ratio (default ν = 0.25)
f_d      - depth factor (default f_d = 1)
```

```
c
      program maxwell
c
c*****************************************************************
c  REFERENCE:
c*****************************************************************
c
c  USAGE:
c  Program to calculate the surface displacement or stress
c  due to planar vertical fault consisting of many elements
c  where the slip varies on each element.  The model consists
c  of an elastic plate of thickness H over a visco-elastic half space.
c  All elements slip between depths D1 and D2 where D1 must be
c  above the base of the plate. One can simulate either a single-couple
c  or double couple dislocation. The displacement or stress
c  can be observed at any depth just above the lower locking depth.
c
c  EXAMPLE:
c  An example of a double-couple fault extending from the base of the
c  plate to a depth of 20 km follows:
c
c*****************************************************************
c  maxwell -50 -50 -20 0 99 1 all_segs.dat 0 xall.grd yall.grd zall.grd
c*****************************************************************
c
c  The three components of surface displacement are stored
c  in three grd-files for use with GMT software.
c
c  Fault elements are stored in the file all_segs.dat
c
c*****************************************************************
c  # dx  sig fault_orientation
c     1. 1.  90.
c  549.36 549.36   0.00 128.53 -40.00   0.00   0.00
c  549.36 549.62 128.53 132.51 -40.00   0.00   0.00
c  549.62 549.88 132.51 136.48 -40.00   0.00   0.00
c  549.88 550.26 136.48 142.45 -40.00   0.00   0.00
c  550.26 551.43 142.45 149.49 -40.00   0.00   0.00
c  551.43 552.47 149.49 154.55 -40.00   0.00   0.00
c*****************************************************************
c
c  The first line is a comment line.
c  The second line has; grid spacing(km), fault thickness(km), and average
c  fault orientation w.r.t the x-axis
c  Subsequent lines have the end points of the fault followed by slip
c  rates; strike-slip, dip-slip, and opening. Millions of elements
c  could be used without a change in memory or execution time.
c
c  A mirror fault is constructed to match the
c  the far-field boundary conditions. For an infinitely-long, straight fault,
c  this program generates the arctangent model.
c
c*****************   main program   ***********************
c
      implicit real*8 (a,b,d-h,o-z)
      implicit complex*16 (c)
      real*8 kx,ky
```

```
c**User: change ni and nj as needed************************
      parameter(ni=2048,nj=2048,nwork=32768,nj2=nj/2+1,ni2=ni/2+1)
c*************************************************************
      parameter (nl=50)
      character*80 felement,fxdisp,fydisp,fzdisp
      character*80 cfac,cd1,cd2,cz,ct,cdbl,cstr
      character*80 cthk,ceta,cpois,cfd
      real*8 An(nl),rla(nl),rmu(nl)
c
      common/plate/rlam1,rlam2,rmu1,rmu2,rho,rc,rd,alph,pi
c
c  the working arrays are all real*4, complex*8 to save memory
c
      real*4 fx(nj,ni),fy(nj,ni),fz(nj,ni)
      real*4 u(nj,ni),v(nj,ni),w(nj,ni)
      complex*8 fkx(nj2,ni),fky(nj2,ni),fkz(nj2,ni)
      complex*8 uk(nj2,ni),vk(nj2,ni),wk(nj2,ni)

      dimension n(2)
      complex*8 work(nwork)
      equivalence (fx(1,1),fkx(1,1))
      equivalence (fy(1,1),fky(1,1))
      equivalence (fz(1,1),fkz(1,1))
      equivalence (u(1,1),uk(1,1))
      equivalence (v(1,1),vk(1,1))
      equivalence (w(1,1),wk(1,1))
c
c*********User: set these default model parameters************

      rho=3300.
      fr=0.6
      young=6.5e10
      pois=0.25
      eta=1.e19
      thk=-50.
      fd=1.0
c*************************************************************
      pi=acos(-1.)
      spyr=86400*365.25
c
c  zero the arrays fx,fy,fz
c
      do 30 i=1,ni
      do 30 j=1,nj
      fx(j,i)=0.
      fy(j,i)=0.
      fz(j,i)=0.
  30  continue
c
c  set the dimensions for fourt
c
      n(1)=nj
      n(2)=ni
c
c   get values from command line
c
      narg = iargc()
      if(narg.lt.11) then
        write(*,'(a)')'  '
        write(*,'(a)')
```

```
     &  'Usage: maxwell F D1 D2 Z T dble ele.dat istr U.grd V.grd W.grd'
        write(*,'(a)')
        write(*,'(a)')
     &  '       F       - factor to multiply output files'
        write(*,'(a)')
     &  '       D1, D2  - depth to bottom and top of fault (neg) '
        write(*,'(a)')
     &  '       Z       - depth of observation plane '
        write(*,'(a)')
     &  '       T       - time since earthquake (yr) '
        write(*,'(a)')
     &  '       dble    - (0-single couple, 1-double couple)'
        write(*,'(a)')
     &  '       ele.dat - file of: x1,x2,y1,y2,F1,F2,F3'
        write(*,'(a)')
     &  '       istress - (0)-disp. U,V,W'
        write(*,'(a)')
     &  '                 (1)-stress Txx,Tyy,Txy'
        write(*,'(a)')
     &  '                 (2)-stress Tnormal,Tshear,Tcoulomb'
        write(*,'(a)')
     &  '       x,y,z.grd - output files of disp. or stress '
        write(*,'(a)')'  '
       stop
       else
         call getarg(1,cfac)
         call getarg(2,cd1)
         call getarg(3,cd2)
         call getarg(4,cz)
         call getarg(5,ct)
         call getarg(6,cdbl)
         call getarg(7,felement)
         call getarg(8,cstr)
         read(cfac,*)fac
         read(cd1,*)d1
         read(cd2,*)d2
         read(cz,*)zobs
         read(ct,*)timey
         time=spyr*timey
           read(cdbl,*)idble
         read(cstr,*)istr

         call getarg(9,fxdisp)
         nc=index(fxdisp,' ')
         fxdisp(nc:nc)='\0'
         call getarg(10,fydisp)
         nc=index(fydisp,' ')
         fydisp(nc:nc)='\0'
         call getarg(11,fzdisp)
         nc=index(fzdisp,' ')
         fzdisp(nc:nc)='\0'
         if(narg.eq.15) then
           call getarg(12,cthk)
           call getarg(13,ceta)
           call getarg(14,cpois)
           call getarg(15,cfd)
           read(cthk,*)thk
           read(ceta,*)eta
           read(cpois,*)pois
```

```
          read(cfd,*)fd
           if(thk.gt.d1.or.d1.gt.d2.or.d2.gt.0.or.
     +     zobs.lt.d1) then
          write(*,'(a)')' depths  must be  negative H < d1 < d2 '
          stop
         endif
c
c   scale d1 by fd for double couple models only
c
           if(idble.eq.1) d1=d1*fd
           write(*,*)thk,eta,pois,d1,d2
         endif
         endif
c
c  open the input file and load the force arrays
c
       open(unit=5,file=felement,status='old')
       read(5,*)cd1
c
c  read the fault constants
c
       read(5,*) dx,sig,psi
c
       width=dx*nj
       height=dx*ni
       if(pois.ge.5.) then
       write(*,'(a)')' error in Poisson ratio '
         stop
       endif
       rlam1=young*pois/((1.+pois)*(1.-2.*pois))
       rmu1=0.5*young/(1.+pois)
       alph=(rlam1+rmu1)/(rlam1+2*rmu1)
       rc=alph/(4.*rmu1)
       rd=(rlam1+3*rmu1)/(rlam1+rmu1)
       sig=sig/dx
       if(sig.lt.1)sig=1
c
c  read the elements
c
   50  read(5,*,end=999)x1,x2,y1,y2,F1,F2,F3
       x1=x1/dx
       x2=x2/dx
c
c  invert the y-positions to allign with i-index
c
       y1=ni-y1/dx
       y2=ni-y2/dx
c
c  figure out the part of the array to add the new forces
c
       ipd=3*sig+2
       jx0=min(x1,x2)-ipd
       jxf=max(x1,x2)+.5+ipd
       iy0=min(y1,y2)-ipd
       iyf=max(y1,y2)+.5+ipd
       do 100 iy=iy0,iyf
```

```
if(jx.le.0.or.jx.gt.nj) go to 200
        if(iy.le.0.or.iy.gt.ni) go to 200
        x=jx
        y=iy
        call element(x,y,x1,y1,x2,y2,dx,sig,idble,F1,F2,F3,fxx,fyy,fzz)
c
c remove the x-forces if the fault is single couple
c
        fx(jx,iy)=fx(jx,iy)+fxx
        fy(jx,iy)=fy(jx,iy)+fyy
        fz(jx,iy)=fz(jx,iy)+fzz
        if(idble.eq.0) fx(jx,iy)=0.0
 200  continue
 100  continue
      go to 50
 999  close(unit=5)
c
c  the source in the x-direction
c  this will make the finite displacement BC's match
c  one could also use a cosine transform
c
      do 150 iy=1,ni
      do 150 jx=1,nj2
      fx(nj+1-jx,iy)=-fx(jx,iy)
      fy(nj+1-jx,iy)= fy(jx,iy)
      fz(nj+1-jx,iy)= fz(jx,iy)
 150  continue
c
c  take the fourier transform of the forces
c
      call fourt(fx,n,2,-1,0,work,nwork)
      call fourt(fy,n,2,-1,0,work,nwork)
      call fourt(fz,n,2,-1,0,work,nwork)
c
c  make the coefficients for the visco-elastic half space
c
      call coefan(eta,time,nl,An,namx,rla,rmu)

c
c  construct the fault solution
c
      do 255 i=1,ni
      ky=-(i-1)/height
      if(i.ge.ni2) ky= (ni-i+1)/height
      do 255 j=1,nj2
      kx=(j-1)/width
      cfkx=fkx(j,i)
      cfky=fky(j,i)
      cfkz=fkz(j,i)
      call fvisco(kx,ky,thk,d1,d2,zobs,An,rla,rmu,namx,
     +                                  cfkx,cfky,cfkz,
     +                                    cuk,cvk,cwk)
c
c  now either output displacement or stress
c
      uk(j,i)=fac*rmu1*cuk/(ni*nj)
      vk(j,i)=fac*rmu1*cvk/(ni*nj)
      wk(j,i)=fac*rmu1*cwk/(ni*nj)
```

```
c  compute the stress if requested.
c
      if(istr.ge.1) then
        cux=cmplx(0.,2.*pi*kx)*uk(j,i)
        cvy=cmplx(0.,2.*pi*ky)*vk(j,i)
        cuy=cmplx(0.,2.*pi*ky)*uk(j,i)
        cvx=cmplx(0.,2.*pi*kx)*vk(j,i)
        cwz=rlam1*(cux+cvy)/(rlam1+2.*rmu1)
c
c  assume units are km so divide by 1000
c  need another factor of 1000 because slip is in mm/yr
c
        uk(j,i)=1.e-6*((rlam1+2*rmu1)*cux+rlam1*(cvy+cwz))
        vk(j,i)=1.e-6*((rlam1+2*rmu1)*cvy+rlam1*(cux+cwz))
        wk(j,i)=1.e-6*(rmu1*(cuy+cvx))
      endif
 255  continue
c
c  do the inverse fft's
c
      call fourt(u,n,2,1,-1,work,nwork)
      call fourt(v,n,2,1,-1,work,nwork)
      call fourt(w,n,2,1,-1,work,nwork)
c
c  compute the coulomb stress if requested
c
      if(istr.eq.2) then
       do 260 i=1,ni
        do 260 j=1,nj
        txx=u(j,i)
        tyy=v(j,i)
        txy=w(j,i)
        call coulomb(txx,tyy,txy,fr,psi,rnorm,rshr,rcoul)
        u(j,i)=rnorm
        v(j,i)=rshr
        w(j,i)=rcoul
 260  continue
      endif
c
c  write 3 grd files
c
      rln0=0.
      rlt0=0.
      ddx=dx
      ddy=dx
      rland=9998.*fac
      rdum=9999.*fac
      if(istr.eq.1) then
       rland=9998.d0*rmu1*fac
       rdum=9999.d0*rmu1*fac
      endif
      call writegrd(u,nj,ni,rln0,rlt0,ddx,ddy,
     +              rland,rdum,fxdisp,fxdisp)
      call writegrd(v,nj,ni,rln0,rlt0,ddx,ddy,
     +              rland,rdum,fydisp,fydisp)
      call writegrd(w,nj,ni,rln0,rlt0,ddx,ddy,
     +              rland,rdum,fzdisp,fzdisp)
      stop
      end
```

```
c
      subroutine element(x,y,x1,y1,x2,y2,dxx,sig,idble,F1,F2,F3,
     +                   fxx,fyy,fzz)
c
      implicit real*8(a-h,o-z)
c
      common/plate/rlam1,rlam2,rmu1,rmu2,rho,rc,rd,alph,pi
c
c       create the force vector for this element
c
c       input:
c
c       x, y    - computation point      (km)
c       x1,y1   - start position         (km)
c       x2,y2   - end position           (km)
c       dxx     - grid spacing           (km)
c       sig     - source width in pixels (try 1)
c       idble   - (0-single couple, 1-double couple)
c       F1      - strike-slip
c       F2      - dip-slip
c       F3      - opening mode
c
c       output:
c
c       fxx     - x-component of force couple
c       fyy     - y-component of force couple
c       fzz     - z-component of force couple
c
c  set the x and y taper lengths
c
      sigy=sig
c
c  sigx  should be = 2 or perhaps 4
c
      sigx=2.0
c
c  compute the length and orientation of the fault
c
      Dx=(x2-x1)
      Dy=(y2-y1)
      L=sqrt(Dx*Dx+Dy*Dy)
      theta=atan2(Dy,Dx)
      st=sin(theta)
      ct=cos(theta)

c
c  rotate the vector into the model space
c
      x0=x-x1
      y0=y-y1
      xp=x0*ct+y0*st
      yp=-x0*st+y0*ct

c  compute the h and dh/dx functions
c
      h=exp(-0.5*((yp/sigy)**2))/(sqrt(2*pi)*sigy)/(dxx*dxx)
      dh=-yp*exp(-0.5*((yp/sigy)**2))/(sqrt(2*pi)*sigy**3)/(dxx*dxx)
```

```
c
c   compute the g and dgdx functions
c
      if(xp .gt. -sigx .and. xp .lt. sigx) then
         g=0.5*(1-cos(pi*(xp+sigx)/(2*sigx)))
         dg=pi*sin(pi*(xp+sigx)/(2*sigx))/(4*sigx)
      else if(xp .ge. sigx .and. xp .le. L-sigx) then
         g=1.
         dg=0.
      else if(xp .gt. L-sigx .and. xp .lt. L+sigx) then
         g=0.5*(1-cos(pi*(L+sigx-xp)/(2*sigx)))
         dg=-pi*sin(pi*(L+sigx-xp)/(2*sigx))/(4*sigx)
      else
         g=0.
         dg=0.
      endif
c
c extract components
c
c  primary couple
c
      fxx=g*dh*(-F1*ct+F3*st)
      fyy=g*dh*(F1*st+F3*ct)
      fzz=g*dh*F2
c
c  optional double couple
c
      if(idble.eq.1) then
        fxx=fxx+dg*h*F1*st
        fyy=fyy+dg*h*F1*ct
      endif
      return
      end
```

```
c
      subroutine coefan(reta,timei,nmax,An,namx,rla,rmu)
c
c  input
c    reta - viscosity (Pa S)
c    time - time (s)
c    nmax - max number of coefficients to compute
c
c  output
c    An   - coefficients
c    namx - number of An coefficients needed
c    rla  - lambda coresponding to each coefficient
c    rmu  - mu corresponding to each coefficient
c
      implicit real*8 (a-h,o-z)
      real*8 reta,timei,An(1),rla(1),rmu(1)
      common/plate/rlam1,rlam2,rmu1,rmu2,rho,rc,rd,alph,pi
c
c scale both a and time to make the exponents smaller
c
      scale=1.e11
      bulk=rlam1+2*rmu1/3.
c
```

```
c
      fac=1.
      time=timei/scale
      a=scale*rmu1/(2.*reta)
      at=a*time
      eat=0.0
      if(at.lt.200) eat=exp(-at)
      An(1)=(1.-eat)
      Bnsave=An(1)/a
c
c  do all of the terms
c
      do 50 n=2,nmax
      if(n.gt.2) fac=fac*(n-1)
      term=-eat*(time**(n-1))/a
      Bn=term+(n-1)*Bnsave/a
      Bnsave=Bn
      An(n)=Bn*(a**n)/fac
c
c  zero terms when less than 1.e-5
c
      if(An(n).le.1.e-05) then
       An(n)=0.
      endif
  50  continue
c
c  compute the corresponding elastic constants
c
      namx=nmax
      do 100 n=1,nmax
      if(namx.eq.nmax.and.An(n).eq.0.) namx=n-1
      c1=An(n)**(1./n)
      rat=(1-c1)/(1+c1)
      rmu(n)=rmu1*rat
      rla(n)=bulk-2.*rmu(n)/3.
 100  continue
      return
      end
```

```
c
       subroutine fvisco(kx,ky,H,d1,d2,z,An,rla,rmu,namx,
     +                                     cfkx,cfky,cfkz,
     +                                       cuk,cvk,cwk)
c
c  input
c     kx  - x wavenumber        (1/km)
c     ky  - y wavenumber        (1/km)
c     H   - thickness of plate  (km)
c     d1  - bottom of fault d1 < d2 < 0. (km)
c     d2  - top of fault        (km)
c     z   - observation plane   (km)
c     An  - image coefficients
c     rla - lambda's
c     rmu - mu's
c     namx - number of non-zero coefficients
c     cfkx,cfky,cfkz - force vector
```

```
c
c  output
c     cuk,cvk,cwk - displacement vector
c
c  internal variables
c     cux,cuy,cuz  - elements of force/displacement matrix
c     cvx,cvy,cvz               (km**2/Pa)
c     cwx,cwy,cwz
c     cx,cy,cz  - non-zero vertical stress at surface z=0
c                              (km)
c
      implicit real*8 (a,b,d-h,o-z)
      implicit complex*16 (c)
      implicit real*8 (k)
      real*8 An(1),rla(1),rmu(1)
c
c  routine to compute the greens function for a fault element
c
      common/plate/rlam1,rlam2,rmu1,rmu2,rho,rc,rd,alph,pi
c
      cuk=cmplx(0.,0.)
      cvk=cmplx(0.,0.)
      cwk=cmplx(0.,0.)
      cTk=cmplx(0.,0.)
      if (kx.eq.0.and.ky.eq.0.) return
c
      H1=abs(H)
      H2=abs(2*H)
c
c  use the wavenumber and the number of coefficients An to limit mmax
c
      beta=sqrt(kx*kx+ky*ky)
      mmax=10./(beta*H2)+5
      mmax=min(mmax,namx)
c
c switch to plate model when rmu(1) is exactly zero
c
      if(rmu(1).eq.0.)mmax=1
c
      do 100 m=-mmax,mmax
      i=abs(m)
c
c   load the coefficients
c
      if(i.eq.0) then
        fac=1.
        rlam2=rlam1
        rmu2=rmu1
      else
        fac=An(i)
        rlam2=rla(i)
        rmu2=rmu(i)
      endif
      sgn=1.0
      if(m.gt.0)sgn=-1.0
c
c
c   compute the sources
c
      if(z.lt.d2.and.m.eq.0) then
```

```
c  need to use two parts for the primary source
c
      dm=z
      call fterm(kx,ky,d1,dm,z,H2,m,ux1,uy1,uz1,
     +                             vy1,vz1,wz1,
     +                             c1x,c1y,c1z)
      call fterm(kx,ky,dm,d2,z,H2,m,ux2,uy2,uz2,
     +                             vy2,vz2,wz2,
     +                             c2x,c2y,c2z)
      uxs=ux1-ux2
      uys=uy1-uy2
      uzs=uz1-uz2
      vys=vy1-vy2
      vzs=vz1-vz2
      wzs=wz1-wz2
      c3x=c1x-c2x
      c3y=c1y-c2y
      c3z=c1z-c2z
      else
      call fterm(kx,ky,d1,d2,z,H2,m,uxs,uys,uzs,
     +                             vys,vzs,wzs,
     +                             c3x,c3y,c3z)
      endif
c
c   compute the images
c
      call fterm(kx,ky,-d1,-d2,z,H2,-m,uxi,uyi,uzi,
     +                               vyi,vzi,wzi,
     +                               c3x,c3y,c3z)
c
      cux=sgn*fac*cmplx(uxs+uxi,0)
      cuy=sgn*fac*cmplx(uys+uyi,0)
      cuz=sgn*fac*cmplx(0,uzs+uzi)
c
      cvx=sgn*fac*cmplx(uys+uyi,0.)
      cvy=sgn*fac*cmplx(vys+vyi,0.)
      cvz=sgn*fac*cmplx(0.,vzs+vzi)
c
      cwx=sgn*fac*cmplx(0.,uzs-uzi)
      cwy=sgn*fac*cmplx(0.,vzs-vzi)
      cwz=sgn*fac*cmplx(wzs-wzi,0.)
c
c  update the displacement
c
      cuk=cuk+(cfkx*cux+cfky*cuy+cfkz*cuz)
      cvk=cvk+(cfkx*cvx+cfky*cvy+cfkz*cvz)
      cwk=cwk+(cfkx*cwx+cfky*cwy+cfkz*cwz)
c
c   compute the unbalanced vertical stress from the primary source and
c   image only
c
      if(m.eq.0) cTk=(cfkx*c3x+cfky*c3y+cfkz*c3z)
c
  100 continue
c   load the coefficients for the Boussinesq problem use rla(1) and rmu(1)
c
      if(namx.eq.0) then
        rlam2=rlam1
        rmu2=rmu1
```

```
else
        rlam2=rla(1)
        rmu2=rmu(1)
      endif
      call boussinesql(kx,ky,z,H1,cub,cvb,cwb,cdwdz)
c
c  compute the total displacement
c
      cuk=cuk+cTk*cub
      cvk=cvk+cTk*cvb
      cwk=cwk+cTk*cwb
c
      return
      end
c
```

```
c
      subroutine fterm(kx,ky,d1,d2,z,H2,m,ux,uy,uz,
     +                                 vy,vz,wz,
     +                                 c3x,c3y,c3z)
c
c  input
c    kx  - x wavenumber        (1/km)
c    ky  - y wavenumber        (1/km)
c    d1  - bottom of fault in source or image (km)
c    d2  - top of fault in source or image (km)
c    z   - observation plane   (km)
c    H2  - 2 x plate thickness
c    m   - image number
c
c  output
c    cux,cuy,cuz  - elements of force/displacement matrix
c    cvy,cvz,cwz               (km**2/Pa)
c    c3x,c3y,c3z  - non-zero vertical stress at surface z=0
c                               (km)
c
      implicit real*8(a,b,d-h,o-z)
      implicit complex*16 (c)
      implicit real*8 (k)
      common/plate/rlam1,rlam2,rmu1,rmu2,rho,rc,rd,alph,pi
c
      if (kx.eq.0.and.ky.eq.0.) return
c
c  generate the wavenumbers
c
      kh2=kx*kx+ky*ky
      kh=sqrt(kh2)
      B=2*pi*kh
      B2=4*pi*pi*kh2
      kxx=(kx*kx)/kh2
      kyy=(ky*ky)/kh2
      kxy=(kx*ky)/kh2
      fac=rc/B2
c
c  generate the coefficients
c
      a01=rd+kyy-kxx
      a02=-kxx
      a03=-2*kxy
```

```
      a04=-kxy
      a05=-kx/kh
      a06=a05
      a07=rd+kxx-kyy
      a08=-kyy
      a09=-ky/kh
      a10=a09
      a11=rd+1.
      a12=1.
c
c  generate and subtract the exponentials
c
      Bs1=B*abs(z-(d1+H2*m))
      Bs2=B*abs(z-(d2+H2*m))
      if(Bs1.gt.50.) then
        es1=0.
      else
        es1=exp(-Bs1)
      endif
      zes1=Bs1*es1
      if(Bs2.gt.50.) then
        es2=0.
      else
        es2=exp(-Bs2)
      endif
      zes2=Bs2*es2
c
c  modify these terms if rmu2 = 0.
c  use the series expansion
c
      if(rmu2.eq.0.0.and.m.ne.0) then
        BH2=B*H2
        if(BH2.gt.50.) then
          exBH2=0.
        else
          exBH2=exp(-BH2)
        endif
        eH2=1.-exBH2
        es1=es1/eH2
        es2=es2/eH2
        zes1=(Bs1+BH2*exBH2/eH2)*es1
        zes2=(Bs2+BH2*exBH2/eH2)*es2
      endif
      es21=es2-es1
      zes21=zes2-zes1
c
c  multiply the coefficients and the exponentials
c
      ux=fac*(a01*es21+a02*zes21)
      uy=fac*(a03*es21+a04*zes21)
      uz=fac*(a05*es21+a06*zes21)
      vy=fac*(a07*es21+a08*zes21)
      vz=fac*(a09*es21+a10*zes21)
      wz=fac*(a11*es21+a12*zes21)
c
c   compute the surface traction components
c
      rl2m=rlam1+2.*rmu1
```

```
c
c  generate the exponentials
c
      Bd1=B*abs(d1+H2*m)
      Bd2=B*abs(d2+H2*m)
      if(Bd1.gt.50) then
        e1=0.
      else
        e1=exp(-Bd1)
      endif
      ze1=Bd1*e1
      if(Bd2.gt.50) then
        e2=0.
      else
        e2=exp(-Bd2)
      endif
      ze2=Bd2*e2
      if(rmu2.eq.0.0.and.m.ne.0) then
        e1=e1/eH2
        e2=e2/eH2
        ze1=(Bd1+BH2*exBH2/eH2)*e1
        ze2=(Bd2+BH2*exBH2/eH2)*e2
      endif
c
c  multiply the coefficients by the exponentials
c
      arg1=(rlam1/rl2m)*(e2-e1)+alph*(ze2-ze1)
      arg2=-(rmu1/rl2m+2*alph)*(e2-e1)-alph*(ze2-ze1)
      c3x=cmplx(0.,arg1*kx/(B*kh))
      c3y=cmplx(0.,arg1*ky/(B*kh))
      c3z=cmplx(arg2/B,0.)
      return
      end
c
```

```
c
      subroutine boussinesql(kx,ky,z,H,cub,cvb,cwb,cdwdz)
c
c  input
c    kx  - x wavenumber        (1/km)
c    ky  - y wavenumber        (1/km)
c    z   - observation plane z < 0   (km)
c    H   - thickness of upper layer (km)
c
c  output
c    cub - x-displacement      (km/Pa)
c    cvb - y-displacement
c    cwb - z-displacement
c    cdwdz - derivative of w with respect to z
c
      implicit real*8 (a,b,d-h,o-z)
      implicit complex*16 (c)
      implicit real*8 (k)
      real*8 c1,c2
c
c  routine to compute the greens function for point load on
c  a layered half-space
c
      common/plate/rlam1,rlam2,rmu1,rmu2,rho,rc,rd,alph,pi
```

```
c
      if (kx.eq.0.and.ky.eq.0.) then
        cub=cmplx(0.,0.)
        cvb=cmplx(0.,0.)
        cwb=cmplx(0.,0.)
        cdwdz=cmplx(0.,0.)
        return
      endif
c
c   compute the coefficients
c
      kh2=kx*kx+ky*ky
      kh=sqrt(kh2)
      B=2*pi*kh
      B2=B*B
      arg=B*z
      if(arg.lt.-50.) then
        ep=0.
        en=exp(50.)
      else
        ep=exp(arg)
        en=exp(-arg)
      endif
      call coefl(rlam1,rlam2,rmu1,rmu2,B,H,rho,a1,b1,c1,d1,a2,c2)
c     write(*,*)rlam1,rlam2,rmu1,rmu2,B,H,rho,a1,b1,c1,d1,a2,c2
c
      if(abs(z).lt.H) then
        arg1=alph*(a1*B*ep+b1*B*en+
     +       c1*(1+B*z)*ep-d1*(1-B*z)*en)
        arg2=alph*(a1*B*ep-b1*B*en+
     +       c1*(2+B*z-2/alph)*ep+d1*(2-B*z-2/alph)*en)
        arg3=0.0

      else
        alph2=(rlam2+rmu2)/(rlam2+2*rmu2)
        arg1=alph2*(a2*B*ep+c2*(1+B*z)*ep)
        arg2=alph2*(a2*B*ep+c2*(2+B*z-2/alph2)*ep)
        arg3=0.0
      endif
      cub=cmplx(0.,-2.*pi*kx*arg1)
      cvb=cmplx(0.,-2.*pi*ky*arg1)
      cwb=cmplx(-B*arg2,0.)
      cdwdz=cmplx(-arg3,0.)
      return
      end
```

```
/* coefl.c */

/* Fortran callable routine to compute the Boussinesq coefficients
for an elastic layer over a half space */

#include <math.h>

int coefl_(l1i,l2i,u1i,u2i,mi,hi,rho,A1,B1,C1,D1,A2,C2)

/* input parameters */

   double *l1i;          /* lambda 1 */
   double *l2i;          /* lambda 2 */
   double *u1i;          /* shear 1 */
   double *u2i;          /* shear 2 */
   double *mi;           /* wavenumber */
   double *hi;           /* layer thickness */
   double *rho;               /* density */

/* output parameters */

   double *A1;           /* coefficient A1 */
   double *B1;           /* coefficient B1 */
   double *C1;           /* coefficient C1 */
   double *D1;           /* coefficient D1 */
   double *A2;           /* coefficient A2 */
   double *C2;           /* coefficient C2 */

{
  double T33,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12;
  double p,g,l1,l2,u1,u2,m,h;
  double arg2,arg4,exp2,exp4;
  double alpha1,mh,denom_common,D11,D22;
  double denom1,denom2,denom_pg,A1x,B1x,C1x,D1x;
  double eta1,alpha2,eta2,m2,mhs,e2,ne2,e4,ne4,emh,Dx1,Dx2;
  double denom_old,denom_new,denom_pgW;

  p=*rho*1.e3;
  g=-9.8;
  T33=1.0;
  l1=*l1i;
  l2=*l2i;
  u1=*u1i;
  u2=*u2i;
  m=*mi;
  h=*hi;

/* branch to appropriate case */

  if (h == 0.) {    /*  halfspace  */

#include "../include/halfspace.h"

  }
  else if(p == 0.) {          /*  layer over halfspace, no gravity */

#include "../include/layered.h"
```

```
  }
  else if(u2 == 0.) {    /*  layer over halfspace, with gravity */

#include "../include/plate.h"

  }
  else {            /*  thick plate with gravity */

#include "../include/layered_pg.h"    /* this has simplified coefficients, fixed exp */

  }

}
```

```
/* halfspace.h */

      *A1=T33*(l1+2.0*u1)*l1/u1/(l1+u1)/(l1+u1)/(m*m*m)/2.0;
      *B1=0.0;
      *C1=-(l1+2.0*u1)*T33/u1/(l1+u1)/(m*m)/2.0;
      *D1=0.0;
      *A2=T33*(l1+2.0*u1)*l1/u1/(l1+u1)/(l1+u1)/(m*m*m)/2.0;
      *C2=-(l1+2.0*u1)*T33/u1/(l1+u1)/(m*m)/2.0;
```

```
/* layered.h */

/*  set up common expressions   */

          alpha1=(l1+u1)/(l1+2*u1);
          eta1=(l1+u1)/(3*l1+4*u1);
          alpha2=(l2+u2)/(l2+2*u2);
          eta2=(l2+u2)/(3*l2+4*u2);

          m2=m*m;
          mh=m*h;
          mhs=mh*mh;

        arg2=-2.*mh;
        arg4=-4.*mh;
        ne2=0.0;
        ne4=0.0;
          if(arg2 > -50.) ne2=exp(arg2);
          if(arg4 > -50.) ne4=exp(arg4);

          Dx1=  u1*u1*(l1+u1)*(l1+u1)*(l2+3*u2)*(ne4-
2*ne2*(1+2*mhs)+1)+u2*u2*(l1*l1)*(4*u1+u2+l2)*(ne4+2*ne2*(1+2*mhs)+1)+u1*u2*(u1*u2*((l2+u2
)*(3*ne4+2*ne2*(5+2*mhs)+3*1)-2*u1*(3*ne4+2*ne2*(1-2*mhs)-5*1)-2*l1*(5*ne4+2*ne2*(1-
4*mhs)-7*1))+4*u2*(l1*(l2+u2)*(ne4+2*ne2*(1+mhs)+1)-2*l1*l1*(ne4+ne2*(1+mhs)))-
2*l2*(l1+u1)*(l1+2*u1)*(ne4-1));

          denom_common=m*m*m*m*(l1+u1)*(l2+u2)/T33/((l1+2*u1)*(l1+2*u1))/((l2+2*u2)*(l2+2*
u2));
          denom_old=-2*u1*m/(l1+2*u1)*Dx1;
          denom_new=denom_common*(denom_old);
```

```
          A1x=   u1*u1*l2*(l1+u1)*(2*u1*mh*mh*ne2-l1*(1-ne2*(1-2*mh+2*mh*mh)))-
u2*u2*l1*(l2+u2)*(l1*(1+ne2*(1-2*mh+2*mh*mh)))-u1*u2*(u1*(4*l1*l2-8*u1*u2*ne2*(1-
mh*mh)+3*(l1+u1)*(l1*(1-ne2*(1-2*mh+2*mh*mh))-2*u1*mh*mh*ne2))+u2*(4*l1*l1*(1-ne2*(mh-
mh*mh))+l1*(l2+u2)*(3+ne2*(3-2*mh+4*mh*mh)))+2*u1*u2*(l1*(5-2*ne2*(1+mh-2*mh*mh))+2*u1*ne2*(1-
mh*mh)+(l2+u2)*ne2*(2+mh*mh))+2*l1*l1*l2);


          *A1=1.0/denom_new*(m*m*(l2+u2)/((l1+2*u1)*(l1+2*u1))/((l2+2*u2)*(l2+2*u2)))*A1x;

          B1x=   u1*u1*l2*(l1+u1)*(l1*(1+2*mh*(1+mh)-ne2)+2*u1*mh*mh)-
u2*u2*l1*(l2+u2)*(l1*(1+2*mh*(1+mh)+ne2))+u1*u2*(6*u1*u1*u1*mh*mh+u1*(3*u1*l1*(1+2*mh*(1+2*mh)-
ne2)+3*l1*l1*(1+2*mh*(1+mh)-ne2)+4*l1*l2*ne2)-
u2*(l1*(l2+u2)*(3+2*mh*(1+2*mh)+3*ne2)+4*l1*l1*(mh*(1+mh)-ne2))+2*u1*u2*(2*u1*(1-mh*mh)+l1*(2-
2*mh*(1+2*mh)+3*ne2)-(l2+u2)*(2+mh*mh))+2*l1*l1*l2*ne2);

          *B1=1.0/denom_new*(ne2*m*m*(l2+u2)/((l1+2*u1)*(l1+2*u1))/((l2+2*u2)*(l2+2*u2)))*B1x;

          C1x=   u1*u1*l2*(l1+u1)*(1-ne2*(1-2*mh))+u2*u2*l1*(l2+u2)*(1+ne2*(1-
2*mh))+u1*u2*(u1*(3*(l1+u1)*(1-ne2*(1-2*mh)))+u2*(2*l1*(2+ne2*(1-2*mh))+(l2+u2)*(3+ne2*(1-
2*mh)))+2*u1*u2*(5+ne2*(1-2*mh))+2*l2*(l1+2*u1));

          *C1=1.0/denom_new*(m*m*m*(l1+u1)*(l2+u2)/((l1+2*u1)*(l1+2*u1))/((l2+2*u2)*(l2+2*u2)))*
C1x;

          D1x=   u1*u1*l2*(l1+u1)*(1+2*mh-ne2)-
u2*u2*l1*(l2+u2)*(1+2*mh+ne2)+u1*u2*(u1*(3*(l1+u1)*(1+2*mh-ne2))-u2*(2*l1*(1+2*mh-
2*ne2)+(l2+u2)*(1+2*mh+3*ne2))-2*u1*u2*(1+2*mh-3*ne2)+2*l2*(l1+2*u1)*ne2);

          *D1=1.0/denom_new*(ne2*m*m*m*m*(l1+u1)*(l2+u2)/((l1+2*u1)*(l1+2*u1))/((l2+2*u2)*(l2+2*u2
)))*D1x;
```

```
/* layerd_pg.h */

/*  set up common expressions   */

          alpha1=(l1+u1)/(l1+2*u1);
          eta1=(l1+u1)/(3*l1+4*u1);
          alpha2=(l2+u2)/(l2+2*u2);
          eta2=(l2+u2)/(3*l2+4*u2);

          m2=m*m;
          mh=m*h;
          mhs=mh*mh;

        arg2=-2.*mh;
        arg4=-4.*mh;
        ne2=0.0;
        ne4=0.0;
          if(arg2 > -50.) ne2=exp(arg2);
          if(arg4 > -50.) ne4=exp(arg4);

          Dx1=  u1*u1*(l1+u1)*(l1+u1)*(l2+3*u2)*(ne4-
2*ne2*(1+2*mhs)+1)+u2*u2*(l1*l1)*(4*u1+u2+l2)*(ne4+2*ne2*(1+2*mhs)+1)+u1*u2*(u1*u2*((l2+u2)*(3*n
e4+2*ne2*(5+2*mhs)+3*1)-2*u1*(3*ne4+2*ne2*(1-2*mhs)-5*1)-2*l1*(5*ne4+2*ne2*(1-4*mhs)-
7*1))+4*u2*(l1*(l2+u2)*(ne4+2*ne2*(1+mhs)+1)-2*l1*l1*(ne4+ne2*(1+mhs)))-
2*l2*(l1+u1)*(l1+2*u1)*(ne4-1));
```

```
          Dx2=   -u1*u1*l2*(l1+u1)*(ne4-4*mh*ne2-1)-u2*u2*l1*(l2+u2)*(ne4+4*mh*ne2-
1)+u1*u2*(u1*(4*l2*(ne4+1)-3*(l1+u1)*(ne4-4*mh*ne2-1))+u2*(4*l1*(ne4-2*mh*ne2+1)-
(l2+u2)*(3*ne4+4*mh*ne2-3*1))+2*u1*u2*(3*ne4-4*mh*ne2+5*1)+2*l1*l2*(ne4+1));

          denom_common=m*m*m*m*(l1+u1)*(l2+u2)/T33/((l1+2*u1)*(l1+2*u1))/((l2+2*u2)*(l2+2*u2))
;
          denom_old=-2*u1*m/(l1+2*u1)*Dx1;
          denom_pgW=p*g*Dx2;
          denom_new=denom_common*(denom_old+denom_pgW);


          A1x=   u1*u1*l2*(l1+u1)*(2*u1*mh*mh*ne2-l1*(1-ne2*(1-2*mh+2*mh*mh)))-
u2*u2*l1*(l2+u2)*(l1*(1+ne2*(1-2*mh+2*mh*mh)))-u1*u2*(u1*(4*l1*l2-8*u1*u2*ne2*(1-
mh*mh)+3*(l1+u1)*(l1*(1-ne2*(1-2*mh+2*mh*mh))-2*u1*mh*mh*ne2))+u2*(4*l1*l1*(1-ne2*(mh-
mh*mh))+l1*(l2+u2)*(3+ne2*(3-2*mh+4*mh*mh)))+2*u1*u2*(l1*(5-2*ne2*(1+mh-2*mh*mh))+2*u1*ne2*(1-
mh*mh)+(l2+u2)*ne2*(2+mh*mh))+2*l1*l1*l2);


          *A1=1.0/denom_new*(m*m*(l2+u2)/((l1+2*u1)*(l1+2*u1))/((l2+2*u2)*(l2+2*u2)))*A1x;
          B1x=   u1*u1*l2*(l1+u1)*(l1*(1+2*mh*(1+mh)-ne2)+2*u1*mh*mh)-
u2*u2*l1*(l2+u2)*(l1*(1+2*mh*(1+mh)+ne2))+u1*u2*(6*u1*u1*u1*mh*mh+u1*(3*u1*l1*(1+2*mh*(1+2*mh)
-ne2)+3*l1*l1*(1+2*mh*(1+mh)-ne2)+4*l1*l2*ne2)-
u2*(l1*(l2+u2)*(3+2*mh*(1+2*mh)+3*ne2)+4*l1*l1*(mh*(1+mh)-ne2))+2*u1*u2*(2*u1*(1-mh*mh)+l1*(2-
2*mh*(1+2*mh)+3*ne2)-(l2+u2)*(2+mh*mh))+2*l1*l1*l2*ne2);


          *B1=1.0/denom_new*(ne2*m*m*(l2+u2)/((l1+2*u1)*(l1+2*u1))/((l2+2*u2)*(l2+2*u2)))*B1x;

          C1x=   u1*u1*l2*(l1+u1)*(1-ne2*(1-2*mh))+u2*u2*l1*(l2+u2)*(1+ne2*(1-
2*mh))+u1*u2*(u1*(3*(l1+u1)*(1-ne2*(1-2*mh)))+u2*(2*l1*(2+ne2*(1-2*mh))+(l2+u2)*(3+ne2*(1-
2*mh)))+2*u1*u2*(5+ne2*(1-2*mh))+2*l2*(l1+2*u1));

          *C1=1.0/denom_new*(m*m*m*(l1+u1)*(l2+u2)/((l1+2*u1)*(l1+2*u1))/((l2+2*u2)*(l2+2*u2))
)*C1x;

          D1x=   u1*u1*l2*(l1+u1)*(1+2*mh-ne2)-
u2*u2*l1*(l2+u2)*(1+2*mh+ne2)+u1*u2*(u1*(3*(l1+u1)*(1+2*mh-ne2))-u2*(2*l1*(1+2*mh-
2*ne2)+(l2+u2)*(1+2*mh+3*ne2))-2*u1*u2*(1+2*mh-3*ne2)+2*l2*(l1+2*u1)*ne2);

          *D1=1.0/denom_new*(ne2*m*m*m*(l1+u1)*(l2+u2)/((l1+2*u1)*(l1+2*u1))/((l2+2*u2)*(l2+2*
u2)))*D1x;
```

```
/* plate.h */

      alpha1 = (l1+u1)/(l1+2.0*u1);
      mh = m*h;
      arg2=-2.*m*h;
      arg4=-4.*m*h;
      exp2=0.;
      exp4=0.;
      if(arg2 > -50.) exp2=exp(arg2);
      if(arg4 > -50.) exp4=exp(arg4);

      denom_common = -m*m*m*m*pow(l1+u1,2.0)/pow(l1+2.0*u1,2.0)/T33;

      D11 = exp4-2.0*exp2*(1.0+2.0*mh*mh)+1.0;
      D22 = exp4-4.0*m*h*exp2-1.0;
```

```
      denom1 = 2.0*u1*alpha1*m*D11;
      denom2 = p*g*D22;

      denom_pg = denom_common*(denom1+denom2);

      A1x = (l1+u1)*(2.0*u1*mh*mh*exp2 -l1*(1-exp2*(1-2*mh+2*mh*mh)));
      *A1 = 1/denom_pg*(m*m/pow(l1+2.0*u1,2.0))*A1x;

      B1x = (l1+u1)*(2*u1*mh*mh+l1*(1+2*mh*(1+mh)-exp2));
      *B1 = 1/denom_pg*(exp2*m*m/(pow(l1+2.0*u1,2.0)))*B1x;

      C1x = (1-exp2*(1-2*mh));
      *C1 = 1/denom_pg*(alpha1*alpha1*m*m*m)*C1x;

      D1x = (1+2*mh-exp2);
      *D1 = 1/denom_pg*(exp2*alpha1*alpha1*m*m*m)*D1x;

      *A2=0.;
      *C2=0.;
```

```
c
      subroutine coulomb(txx,tyy,txy,fr,pssi,rn,rs,rcl)
c
c  input
c    txx - xx-component of stress (Pa)
c    tyy - yy-component of stress (Pa)
c    txy - xy-component of stress (Pa)
c    tzz - zz-component of stress (Pa)
c    fr  - coefficient of friction
c    psi - fault orientation CCW from east (deg)
c
c  output
c    rn   -  normal stress on fault
c    rs   -  shear stress on fault
c    rcl  -  coulomb stress in Pa
c
      implicit real*8(a-h,o-z)
      common/plate/rlam1,rlam2,rmu1,rmu2,rho,rc,rd,alph,pi
c
      rad=pi/180.
      sp=sin(pssi*rad)
      s2p=sin(2.*pssi*rad)
      cp=cos(pssi*rad)
      c2p=cos(2.*pssi*rad)
      rn=txx*sp*sp-2*txy*sp*cp+tyy*cp*cp
      rs=0.5*(tyy-txx)*s2p+txy*c2p
c
c  compute the coulomb stress
c
c  note, normal stress is taken as negative, so subtracting normal stress
c  leads to adding:  shear + normal  (change 10/8/03)

      rcl=rs+fr*rn
      return
      end
c
```