GEOLOGICAL SURVEY OF CANADA

OPEN FILE 3665

Computing the electromagnetic fields of dipole sources in an anisotropic layered earth

D.E. Boerner

1999

Natural Resources Canada Ressources naturelles Canada

Canada

GEOLOGICAL SURVEY OF CANADA

OPEN FILE 3707

Computing the electromagnetic fields of dipole sources in an anisotropic layered earth

D.E. Boerner

Continental Geoscience Division
Geological Survey of Canada
615 Booth Street
Ottawa, Ontario, CANADA K1Y 0E9

1999

COMPUTING THE ELECTROMAGNETIC FIELDS OF DIPOLE SOURCES IN AN ANISOTROPIC LAYERED EARTH

David E. Boerner
Continental Geoscience Division
Geological Survey of Canada
615 Booth Street, Ottawa, Ontario, CANADA K1A 0E9

Introduction

A generalized representation of the EM fields due to an arbitrary source in a layered earth is presented in terms of toroidal and poloidal modes from an arbitrary dipole source situated in a whole space. Section 2 presents a discussion on how the modal representation can be extended to account for stratified media and it is shown how propagator matrices (see, for example, Kennett 1983) can be used to express the modal potentials at any point in a layered earth (Boerner & West, 1989; note that this paper contains some typographical errors). This result is then used to derive an algorithm useful for computing the EM fields from an arbitrarily oriented point source (electric or magnetic dipole), located at any position in a stratified earth.

1 A Modal Description of EM Fields

Consider the standard form of Maxwell's equations and a single Fourier component proportional to $e^{i\omega t}$,

$$\nabla \cdot \mathbf{B} = 0 \tag{1}$$

$$\nabla \times \mathbf{E} + i\omega\mathbf{B} = 0 \tag{2}$$

$$\nabla \times \mathbf{B} - \mu\alpha\mathbf{E} = \mu\mathbf{J}'. \tag{3}$$

$\mathbf{E}$ is the electric field, $\mathbf{B}$ is the magnetic induction and $\mathbf{J}'$ is the applied source current which is assumed to be unaffected by $\mathbf{E}$ and $\mathbf{B}$. The electrical properties of the uniform and isotropic medium are represented by the magnetic permeability μ and the admittivity $\alpha = \sigma + i\omega\epsilon$, where σ is the electrical conductivity and ϵ is the permittivity. For the development of the layered earth resposne, it is assumed that the spatial variation of these parameters is confined to the z direction. Cylindrical and Cartesian coordinates will be used in the development with a common origin and z-axis and with r, x and y in directions parallel to layering.

Our goal is to derive a solution to Maxwell's equations (1)–(3) in terms of two independent modes. The axis of separation for these modes is chosen to be the direction in which the model parameters vary, *i.e.,* the z-axis. The "Toroidal Magnetic" or TM mode is characterized by current loops in the r-z planes and a toroidal magnetic field (*i.e.,* a solenoidal field which has no z-component). Since the electric currents associated with this mode cut across the changing medium properties, one can expect that

TM modes are sensitive to the concentration of charge on conductivity gradients. The "Poloidal Magnetic" or PM mode consists of current loops lying perpendicular to the z-axis which generate a poloidal magnetic field (*i.e.,* a solenoidal field whose curl has no z-component). PM mode currents are coupled by induction and are sensitive to layers of high conductivity in the medium, rather than to conductivity gradients.

The modal solution to Maxwell's equations can be obtained by employing a standard theorem of vector analysis to decompose any vector field into a combination of three scalar fields (cf. Morse & Feshbach 1953, Chapter 13),

$$\mathbf{F} = \nabla\phi + \nabla \times (\psi\hat{\mathbf{z}}) + \nabla \times \nabla \times (\chi\hat{\mathbf{z}}) \tag{4}$$

where ϕ, ψ, and χ are eigenfunction solutions of the scalar equations $\nabla^2\phi + k^2\phi = 0$, etc. When this decomposition is applied to the magnetic induction, the fact that $\mathbf{B}$ must be solenoidal requires a representation of the form

$$\mathbf{B} = \nabla \times (\Pi\hat{\mathbf{z}}) + \nabla \times \nabla \times (\Psi\hat{\mathbf{z}}), \tag{5}$$

where Π represents a scalar potential which generates TM modes and Ψ is the scalar potential associated with the PM modes. The common Hertz potential separation used by Weaver (1970), among others, is obtained by replacing Π with $\alpha\Pi$. In fact, Π and Ψ are not uniquely determined by this representation until a further constraint is applied (see Backus 1986 §1). In our case, an appropriate constraint is applied when the source current is specified by vertical, horizontally irrotational and divergenceless components,

$$\mathbf{J}' = J'_z\hat{\mathbf{z}} + \nabla_h T + \nabla \times (\Upsilon\hat{\mathbf{z}}) \tag{6}$$

where the subscript h refers to the horizontal components, and the functions T and Υ must satisfy the Poisson equations

$$\nabla_h^2 T = \nabla_h \cdot \mathbf{J}'_h \tag{7}$$

$$\nabla_h^2 \Upsilon = -(\nabla_h \times \mathbf{J}'_h) \cdot \hat{\mathbf{z}}. \tag{8}$$

It is important to point out that there are conditions on T and Υ (see Backus 1986 §5.2), but these do not usually introduce problems when dealing with controlled sources. From Maxwell's equations (1)–(3) and the vector representations in (5)–(6), we can obtain differential equations for Π and Ψ.

$$\nabla_h^2\Pi + \alpha\partial_z(\partial_z\Pi/\alpha) - i\omega\mu\alpha\Pi = -\mu J'_z + \mu\alpha\partial_z(T/\alpha) \tag{9}$$

$$\nabla^2\Psi - i\omega\mu\alpha\Psi = -\mu\Upsilon \tag{10}$$

Also, the electric field is

$$\mathbf{E} = \frac{1}{\alpha}\nabla_h\left[\partial_z\Pi/\mu - T\right] - \frac{1}{\alpha}(\nabla_h^2\Pi/\mu + J'_z)\hat{\mathbf{z}} - i\omega\nabla \times (\Psi\hat{\mathbf{z}}) \tag{11}$$

Equations (9) and (10) illustrate the assertions made in the introduction to this section concerning the current distributions which produce the TM and PM modes. Namely, that divergenceless, horizontally circulating source currents $\nabla \times (\Upsilon\,\hat{\mathbf{z}})$ produce the PM modes, while J'_z and $\nabla_h T$ generate the TM modes.

The next step is to find solutions for Π and Ψ which satisfy the differential equations given above. In particular, we are concerned with point sources which suggests that (9) and (10) may be conveniently solved using the Hankel transform pair,

$$\begin{aligned} \tilde{f}(\lambda, z) &= \int_0^\infty r J_0(\lambda r) f(r, z) dr, \\ f(r, z) &= \int_0^\infty \lambda J_0(\lambda r) \tilde{f}(\lambda, z)\, d\lambda. \end{aligned} \tag{12}$$

$J_0(\lambda r)$ is the zeroth order Bessel function of the first kind, r is the horizontal separation and λ is the horizontal wavenumber. There is a close relationship between (12) and a two dimensional (2D) Fourier transform where $\lambda^2 = p^2 + q^2$. The 1D Hankel transforms are preferred for problems involving point sources to reduce the numerical effort of transformation to the space domain by exploiting known cylindrical symmetries of the resulting fields.

Applying the Hankel transform to the governing differential equations yields

$$\alpha \partial_z (\partial_z \tilde{\Pi}/\alpha) - u^2 \tilde{\Pi} = -\mu \tilde{J}'_z + \mu\alpha \partial_z (\tilde{T}/\alpha) \tag{13}$$

and

$$\partial_z (\partial_z \tilde{\Psi}) - u^2 \tilde{\Psi} = -\mu \tilde{\Upsilon}, \tag{14}$$

where $u^2 = \lambda^2 + i\omega\mu\alpha$, and a tilde denotes variables expressed in the Hankel transform domain. To solve these differential equations requires a knowledge of the specific source term, and for the purposes of this discussion we shall consider the source to be an arbitrarily oriented electric dipole of moment $J = I(dx', dy', dz')$ at an arbitrary point in the model. By solving (13) and (14) using the usual variation of parameters method for each source orientation (see also, Chave & Cox 1982), we find the potential on either side of the source as

$$\begin{bmatrix} \Pi(z|_{z \le z'}) \\ \Pi(z|_{z \ge z'}) \end{bmatrix} = \frac{\mu I}{4\pi} \int_0^\infty \underset{\sim}{\widetilde{\mathbf{C}}} \begin{bmatrix} dx' & dy' & dz' \\ -dx' & -dy' & dz' \end{bmatrix} \begin{bmatrix} \partial_x \\ \partial_y \\ \dfrac{\lambda^2}{u} \end{bmatrix} \frac{J_0(\lambda\xi)}{\lambda} d\lambda \tag{15}$$

and

$$\begin{bmatrix} \Psi(z|_{z \le z'}) \\ \Psi(z|_{z \ge z'}) \end{bmatrix} = \frac{\mu I}{4\pi} \int_0^\infty \underset{\sim}{\widetilde{\mathbf{C}}} \begin{bmatrix} dx' & -dy' & dz' \\ dx' & -dy' & dz' \end{bmatrix} \begin{bmatrix} \partial_y \\ \partial_x \\ 0 \end{bmatrix} \frac{J_0(\lambda\xi)}{u\lambda} d\lambda, \tag{16}$$

where $\underset{\sim}{\widetilde{\mathbf{C}}}$ is a propagator matrix (see, for example, Gilbert & Backus 1969, Kennett 1983, and Ursin 1983) describing the continuation of the potentials away from the source level,

$$\underset{\sim}{\widetilde{\mathbf{C}}} = \begin{bmatrix} e^{u(z-z')} & 0 \\ 0 & e^{-u(z-z')} \end{bmatrix}. \tag{17}$$

This matrix is based on the fact that the potentials must obey the scalar homogeneous Helmholtz equation everywhere except at the source. Although the concept of propagator matrices is likely to be more familar to seismologists than EM researchers, the technique is useful to keep the expressions for the potentials manageable in the case of complicated layering. This formulation also simplifies programming due to modularity.

The horizontal separation between the receiver location (r, ϕ, z) and the source point (r', ϕ', z') is given by the law of cosines

$$\xi = \sqrt{r^2 + r'^2 - 2rr'\cos(\phi - \phi')}. \tag{18}$$

Notice that the EM fields in a 1D earth exhibit translational and rotational invariance, i.e., only the relative separation and orientation between the source and receiver is important and the absolute horizontal reference position is unnecessary. In fact, the only lateral reference point in a layered earth is at infinity where the potentials must approach zero.

Expressions (15) and (16) can be used to find the modal potentials at (r, ϕ, z) in a uniform whole space from a source at (r', ϕ', z'). We next extend the theory to account for layering in the earth model.

2 Propagator Matrices for Stratified Media

Finding the potentials in a stratified space requires relatively simple modifications of the theory outlined in section 1 in that only the propagator matrices must be changed. To begin, we adopt a right handed coordinate system with the positive z axis directed downwards. Imaginary boundaries are inserted in the model at the source and receiver level and layers are numbered in increasing order away from the source level. This potentially confusing notation is resolved by denoting any quantities on the same side of the source as the receiver with a superscript asterisk (*).

The modal potentials in a stratified halfspace are found by combining the potentials for a source in a whole space in (15)-(16) with solutions of homogeneous forms of the differential equations (13) and (14) to satisfy the boundary conditions at interfaces. Essentially, we add to the whole space potential from the source upward and downward "secondary" potentials whose amplitudes are determined by the property contrasts at boundaries in the media. The net effect is to introduce inward travelling components to the outward components (15)-(16) and alter the outward travelling amplitudes. A convenient manner of representing the effect is with a *reflection ratio*, which at any level in the medium is the ratio of the potential which is travelling towards the source (purely reflected potentials) to the potential travelling away from the source (primary and reflected potentials). With this definition and noting that the secondary potentials must propagate unchanged through the source layer, the total potential at the source level can be written in terms of the reflection ratio on either side of the source. The result expressed in the Hankel domain is

$$\begin{bmatrix} \widetilde{\Pi}^*_{out}(z') \\ \widetilde{\Pi}^*_{in}(z') \end{bmatrix} = \underset{\sim}{\widetilde{\mathcal{R}}}^{\Pi} \begin{bmatrix} \widetilde{\Pi}(z|_{z=z'+\epsilon}) \\ \widetilde{\Pi}(z|_{z=z'-\epsilon}) \end{bmatrix} \qquad \text{and} \qquad \begin{bmatrix} \widetilde{\Psi}^*_{out}(z') \\ \widetilde{\Psi}^*_{in}(z') \end{bmatrix} = \underset{\sim}{\widetilde{\mathcal{R}}}^{\Psi} \begin{bmatrix} \widetilde{\Psi}(z|_{z=z'+\epsilon}) \\ \widetilde{\Psi}(z|_{z=z'-\epsilon}) \end{bmatrix}$$

where ε is infinitesimally small and

$$\widetilde{\underset{\sim}{\mathcal{R}}}^{\Pi} = \frac{1}{1 - \widetilde{R}_1^{\Pi}\widetilde{R}_1^{\Pi *}} \begin{bmatrix} 1 & \widetilde{R}_1^{\Pi} \\ \widetilde{R}_1^{\Pi *} & \widetilde{R}_1^{\Pi}\widetilde{R}_1^{\Pi *} \end{bmatrix}, \tag{19}$$

and

$$\widetilde{\underset{\sim}{\mathcal{R}}}^{\Psi} = \frac{1}{1 - \widetilde{R}_1^{\Psi}\widetilde{R}_1^{\Psi *}} \begin{bmatrix} 1 & \widetilde{R}_1^{\Psi} \\ \widetilde{R}_1^{\Psi *} & \widetilde{R}_1^{\Psi}\widetilde{R}_1^{\Psi *} \end{bmatrix}. \tag{20}$$

With no stratification, there are no reflected potentials and thus the reflection ratios are all zero. In this case, (19) and (20) reduce to unity and the whole space potentials are unaltered by this component of the propagation matrices.

Expressions for the reflection ratios can then be found by translating the boundary conditions on the EM fields to boundary conditions on the modal potentials. The relationship between the ratios defined in adjacent layers for the TM mode is given by

$$\widetilde{R}_i^{\Pi} = \left[\frac{\widetilde{X}_i^{\Pi}(\widetilde{R}_{i+1}^{\Pi} + 1) + \widetilde{Y}_i^{\Pi}(\widetilde{R}_{i+1}^{\Pi} - 1)}{\widetilde{X}_i^{\Pi}(\widetilde{R}_{i+1}^{\Pi} + 1) - \widetilde{Y}_i^{\Pi}(\widetilde{R}_{i+1}^{\Pi} - 1)} \right] \mathrm{e}^{-2u_i t_i} \tag{21}$$

where the i^{th} reflection ratio is evaluated at the boundary between the $i^{th} - 1$ and i^{th} layer, t_i is the thickness of the i^{th} layer and

$$\widetilde{X}_i^{\Pi} = \frac{u_i}{u_{i+1}}, \quad \widetilde{Y}_i^{\Pi} = \frac{\alpha_i}{\alpha_{i+1}}.$$

The PM mode reflection ratio has an identical form to (21),

$$\widetilde{R}_i^{\Psi} = \left[\frac{\widetilde{X}_i^{\Psi}(\widetilde{R}_{i+1}^{\Psi} + 1) + \widetilde{Y}_i^{\Psi}(\widetilde{R}_{i+1}^{\Psi} - 1)}{\widetilde{X}_i^{\Psi}(\widetilde{R}_{i+1}^{\Psi} + 1) - \widetilde{Y}_i^{\Psi}(\widetilde{R}_{i+1}^{\Psi} - 1)} \right] \mathrm{e}^{-2u_i t_i} \tag{22}$$

except that

$$\widetilde{X}_i^{\Psi} = \frac{u_i}{u_{i+1}}, \quad \widetilde{Y}_i^{\Psi} = \frac{\beta_i}{\beta_{i+1}}$$

where $\beta_i = i\omega\mu_i$ is the impedivity of the i^{th} layer.

The form of (21) and (22) indicates that reflection ratios can be evaluated recursively. The starting point for the recursion is anywhere beyond the stratification since then there are no subsequent layers to reflect the potential back towards the source and $R_n = 0$. Thus a calculation is performed on each side of the source, starting at the terminating halfspace and continuing in to the source horizon until $\tilde{R}_1$ and $\tilde{R}_1^*$ (the reflection coefficients on both sides of the source level) are each known.

Once the total potential at the source level is known, the potentials at some other depth are found by using a propagator matrix similar to that described for the whole space model. The form of this matrix for the i^{th} layer is

$$\widetilde{\underset{\sim}{\mathrm{C}}}_i = \begin{bmatrix} \mathrm{e}^{-u_i t_i} & 0 \\ 0 & \mathrm{e}^{+u_i t_i} \end{bmatrix} \tag{23}$$

and it describes the attenuation and amplification of the outward and inward travelling potentials, respectively, as well as their respective phase shifts. However, as this matrix only represents a continuation of the potentials through homogeneous material, and it is necessary to enforce the boundary conditions on the potentials at the interfaces with an appropriate boundary condition matrix. These are

$$\underset{\sim}{\widetilde{\mathbf{B}}}_i^{\Pi} = \frac{1}{2}\begin{bmatrix} (\widetilde{Y}_i^{\Pi} + \widetilde{X}_i^{\Pi}) & (\widetilde{Y}_i^{\Pi} - \widetilde{X}_i^{\Pi}) \\ \widetilde{R}_{i+1}^{\Pi}(\widetilde{Y}_i^{\Pi} + \widetilde{X}_i^{\Pi}) & \widetilde{R}_{i+1}^{\Pi}(\widetilde{Y}_i^{\Pi} - \widetilde{X}_i^{\Pi}) \end{bmatrix}, \tag{24}$$

and

$$\underset{\sim}{\widetilde{\mathbf{B}}}_i^{\Psi} = \frac{1}{2}\begin{bmatrix} (\widetilde{Y}_i^{\Psi} + \widetilde{X}_i^{\Psi}) & (\widetilde{Y}_i^{\Psi} - \widetilde{X}_i^{\Psi}) \\ \widetilde{R}_{i+1}^{\Psi}(\widetilde{Y}_i^{\Psi} + \widetilde{X}_i^{\Psi}) & \widetilde{R}_{i+1}^{\Psi}(\widetilde{Y}_i^{\Psi} - \widetilde{X}_i^{\Psi}) \end{bmatrix}. \tag{25}$$

when propagating the potentials from the i^{th} to the $i^{th}+1$ layer.

The final response matrices for converting the primary source potential at the source level to inward and outward propagating potentials at the receiver level are denoted $\underset{\sim}{\widetilde{\mathbf{T}}}$ and $\underset{\sim}{\widetilde{\mathbf{P}}}$ for the toroidal and poloidal magnetic modes and have the form

$$\underset{\sim}{\widetilde{\mathbf{P}}} = \underset{\sim}{\widetilde{\mathbf{C}}}_R^{\Pi} \times \underset{\sim}{\widetilde{\mathbf{B}}}_{R-1}^{\Pi} \times \underset{\sim}{\widetilde{\mathbf{C}}}_{R-1}^{\Pi} \times \ldots \times \underset{\sim}{\widetilde{\mathbf{B}}}_S^{\Pi} \times \underset{\sim}{\widetilde{\mathbf{C}}}_S^{\Pi} \times \underset{\sim}{\widetilde{\mathcal{R}}}^{\Pi}, \tag{26}$$

and

$$\underset{\sim}{\widetilde{\mathbf{T}}} = \underset{\sim}{\widetilde{\mathbf{C}}}_R^{\Psi} \times \underset{\sim}{\widetilde{\mathbf{B}}}_{R-1}^{\Psi} \times \underset{\sim}{\widetilde{\mathbf{C}}}_{R-1}^{\Psi} \times \ldots \times \underset{\sim}{\widetilde{\mathbf{B}}}_S^{\Psi} \times \underset{\sim}{\widetilde{\mathbf{C}}}_S^{\Psi} \times \underset{\sim}{\widetilde{\mathcal{R}}}^{\Psi}, \tag{27}$$

where the layers containing the source and receiver are designated by the subscripts S and R respectively.

Summarizing, the potentials at any receiver location can be calculated from

$$\begin{bmatrix} \Pi_{out}^*(z) \\ \Pi_{in}^*(z) \end{bmatrix} = \frac{\mu I}{4\pi}\int_0^{\infty} \underset{\sim}{\widetilde{\mathbf{P}}} \begin{bmatrix} dx' & dy' & dz' \\ -dx' & -dy' & dz' \end{bmatrix} \begin{bmatrix} \partial_x \\ \partial_y \\ \dfrac{\lambda^2}{u} \end{bmatrix} \frac{J_0(\lambda\xi)}{\lambda} d\lambda \tag{28}$$

and

$$\begin{bmatrix} \Psi_{out}^*(z) \\ \Psi_{in}^*(z) \end{bmatrix} = \frac{\mu I}{4\pi}\int_0^{\infty} \underset{\sim}{\widetilde{\mathbf{T}}} \begin{bmatrix} dx' & -dy' & dz' \\ dx' & -dy' & dz' \end{bmatrix} \begin{bmatrix} \partial_y \\ \partial_x \\ 0 \end{bmatrix} \frac{J_0(\lambda\xi)}{u\lambda} d\lambda. \tag{29}$$

The sum of the outgoing and ingoing modal potentials can be substituted into the definitions of the EM fields (11) and (5) to obtain expressions for the electric field and magnetic induction from a dipole source. After some algebra and collecting terms according to the the spatial derivatives, a general representation of the EM fields from

an arbitrary electric dipole source is written

$$\begin{bmatrix} E_x \\ E_y \\ E_z \\ B_x \\ B_y \\ B_z \end{bmatrix} = \frac{I}{4\pi} \int_0^\infty \underset{\sim}{\widetilde{\mathcal{F}}} \begin{bmatrix} \lambda^{-2}\partial_{xx} \\ \lambda^{-2}\partial_{xy} \\ \partial_x \\ \partial_y \\ 1 \end{bmatrix} \lambda J_0(\lambda\xi)\, d\lambda \tag{30}$$

where $\underset{\sim}{\widetilde{\mathcal{F}}}$ is a (6×5) matrix provided in Table 1 (which is actually for a transversely isotropic medium; see below). Notice that $\underset{\sim}{\widetilde{\mathcal{F}}}$ contains all the information about the electrical properties of the layered earth for an arbitrary dipole source excitation and the geometrical variations of the fields are contained in the (1×5) matrix of partial differentials.

A useful alternative representation is to employ Bessel function relationships to define the geometrical operators in terms of sines and cosines instead of differentials. For example,

$$\begin{bmatrix} \partial_{xx} \\ \partial_{xy} \\ \partial_x \\ \partial_y \\ 1 \end{bmatrix} J_0(\lambda\xi) \Longleftrightarrow \begin{bmatrix} -\dfrac{\cos 2\theta + 1}{2}\lambda^2 & \cos 2\theta \dfrac{\lambda}{\xi} \\ -\dfrac{\sin 2\theta}{2}\lambda^2 & \sin 2\theta \dfrac{\lambda}{\xi} \\ 0 & -\lambda\cos\theta \\ 0 & -\lambda\sin\theta \\ 1 & 0 \end{bmatrix} \begin{bmatrix} J_0(\lambda\xi) \\ J_1(\lambda\xi) \end{bmatrix} \tag{31}$$

and θ is measured from the x-axis.

The final expression for the fields in terms of the geometrical functions can be

written

$$\begin{bmatrix} E_x \\ E_y \\ E_z \\ B_x \\ B_y \\ B_z \end{bmatrix} = \frac{I}{4\pi} \int_0^\infty \widetilde{\underset{\sim}{\mathcal{F}}} \begin{bmatrix} -\dfrac{\cos 2\theta + 1}{2} & \dfrac{\cos 2\theta}{\lambda\xi} \\ -\dfrac{\sin 2\theta}{2} & \dfrac{\sin 2\theta}{\lambda\xi} \\ 0 & -\lambda\cos\theta \\ 0 & -\lambda\sin\theta \\ 1 & 0 \end{bmatrix} \lambda \begin{bmatrix} J_0(\lambda\xi) \\ J_1(\lambda\xi) \end{bmatrix} d\lambda \tag{32}$$

Table 1 contains the matrix $\widetilde{\underset{\sim}{\mathcal{F}}}$ for two types of sources, the electric dipole source developed above, and the magnetic dipole source $\mathbf{M}$, (*i.e.,* a divergenceless source current). Provided the source is infinitesimally small, altering the theory to account for a magnetic dipole source is relatively simple since the source terms only alter the differential equations (13)–(14) and whole space potentials (15)–(16), but not the propagation matrices. The subscripts S and R in Table 1 refer to the properties of the layers containing the source and receiver, respectively, while the terms $\widetilde{P}_n$ and $\widetilde{T}_n$ refer to linear combinations of the elements of the propagation matrix $\widetilde{\underset{\sim}{\mathbf{P}}}$ and $\widetilde{\underset{\sim}{\mathbf{T}}}$, viz.,

$$\begin{aligned}
\widetilde{T}_1 &= (\widetilde{T}_{1,1} + \widetilde{T}_{1,2} + \widetilde{T}_{2,1} + \widetilde{T}_{2,2}) \\
\widetilde{T}_2 &= (\widetilde{T}_{1,1} + \widetilde{T}_{1,2} - \widetilde{T}_{2,1} - \widetilde{T}_{2,2}) \\
\widetilde{T}_3 &= (\widetilde{T}_{1,1} - \widetilde{T}_{1,2} - \widetilde{T}_{2,1} + \widetilde{T}_{2,2}) \\
\widetilde{T}_4 &= (\widetilde{T}_{1,1} - \widetilde{T}_{1,2} + \widetilde{T}_{2,1} - \widetilde{T}_{2,2}) \\
\widetilde{P}_1 &= (\widetilde{P}_{1,1} + \widetilde{P}_{1,2} + \widetilde{P}_{2,1} + \widetilde{P}_{2,2}) \\
\widetilde{P}_2 &= (\widetilde{P}_{1,1} + \widetilde{P}_{1,2} - \widetilde{P}_{2,1} - \widetilde{P}_{2,2}) \\
\widetilde{P}_3 &= (\widetilde{P}_{1,1} - \widetilde{P}_{1,2} - \widetilde{P}_{2,1} + \widetilde{P}_{2,2}) \\
\widetilde{P}_4 &= (\widetilde{P}_{1,1} - \widetilde{P}_{1,2} + \widetilde{P}_{2,1} - \widetilde{P}_{2,2}).
\end{aligned} \tag{33}$$

Examining Table 1 indicates that once the source orientation is specified, a maximum of only four of these scalar functions are necessary to represent all the components of the EM field. Furthermore, if the source and receiver are on the same horizontal plane this number reduces to two and there is substantial redundancy in how EM field components sample a stratified earth structure.

Equation (30) is a complete representation of the EM Green's function multiplied by the source distribution and no approximations or assumptions (other than choosing

the specific nature of the dipole source) have been incorporated into the development. Because no quasi-static approximation (*e.g.*, Wait 1982) has been made, this form of the EM fields is equally valid for wave and diffusion problems (assuming a known source current that is unaffected by **E** and **B**).

When the source and receiver are both located at the earth/air interface, the propagation matrices reduce to

$$\underset{\sim}{\widetilde{\mathbf{P}}} = \begin{bmatrix} 1 & 0 \\ \widetilde{R}_1^{\Pi} & 0 \end{bmatrix} \tag{34}$$

and

$$\underset{\sim}{\widetilde{\mathbf{T}}} = \begin{bmatrix} 1 & 0 \\ \widetilde{R}_1^{\Psi} & 0 \end{bmatrix}. \tag{35}$$

Further simplification can be achieved when the source and receiver are located at the air/earth contact and the earth is a uniform halfspace. In this case we write

$$\widetilde{R}_0^{\Pi} = \begin{bmatrix} \frac{\lambda}{u} - \frac{\alpha_0}{\alpha} \\ \frac{\lambda}{u} + \frac{\alpha_0}{\alpha} \end{bmatrix} \tag{36}$$

$$\widetilde{R}_0^{\Psi} = \begin{bmatrix} \frac{\lambda}{u} - \frac{\beta_0}{\beta} \\ \frac{\lambda}{u} + \frac{\beta_0}{\beta} \end{bmatrix} \tag{37}$$

since $\widetilde{R}_2^{\Pi} = 0$ and $\widetilde{R}_2^{\Psi} = 0$. For this particular case, it is a simple matter to write down the expressions for $\underset{\sim}{\widetilde{\mathbf{P}}}$ and $\underset{\sim}{\widetilde{\mathbf{T}}}$.

$$\widetilde{P}_1 = \widetilde{P}_4 = 1 + \widetilde{R}_0^{\Pi} = \frac{2\lambda\alpha}{\lambda\alpha + u\alpha_0} \tag{38}$$

$$\widetilde{P}_2 = \widetilde{P}_3 = 1 - \widetilde{R}_0^{\Pi} = \frac{2u\alpha_0}{\lambda\alpha + u\alpha_0} \tag{39}$$

$$\widetilde{T}_1 = \widetilde{T}_4 = 1 + \widetilde{R}_0^{\Psi} = \frac{2\lambda\beta}{\lambda\beta + u\beta_0} \tag{40}$$

$$\widetilde{T}_2 = \widetilde{T}_3 = 1 - \widetilde{R}_0^{\Psi} = \frac{2u\beta_0}{\lambda\beta + u\beta_0} \tag{41}$$

In evaluating closed forms of these expressions, it is useful to recall that $\alpha_0 \approx 0$.

A useful extension to the theory outlined here is to consider transversely isotropic media (see, for example, Wait 1982). That is, the medium is characterized by an admittivity tensor having the principal axis of anisotropy coincident with the axis of separation

for the modal potentials.

$$\underset{\sim}{\alpha} = \begin{bmatrix} \sigma_h + i\omega\epsilon_h & 0 & 0 \\ 0 & \sigma_h + i\omega\epsilon_h & 0 \\ 0 & 0 & \sigma_v + i\omega\epsilon_v \end{bmatrix} = \begin{bmatrix} \alpha_h & 0 & 0 \\ 0 & \alpha_h & 0 \\ 0 & 0 & \alpha_v \end{bmatrix} \tag{42}$$

Thus, the horizontal admittivity may be different than the vertical admittivity.

The development of the modal potentials and EM fields in anisotropic media is more involved than the theory presented here but contains the same essential elements (see the discussion by Nobes 1984). Rather than present a complete theoretical exposition, we simply quote the changes to the theory presented in this paper which are required to represent the EM fields in an anisotropic media.

First, the coefficient of anisotropy is defined to be

$$K^2 = \alpha_h/\alpha_v \tag{43}$$

and it is found that

$$v = \sqrt{K^2\lambda^2 + i\omega\mu\alpha_h} \tag{44}$$

describes the vertical continuation operator for the TM potential in the transversely isotropic media. Therefore the continuation operator for this mode is simply

$$\underset{\sim}{\widetilde{C}}_i^{\Pi} = \begin{bmatrix} e^{-v_i t_i} & 0 \\ 0 & e^{+v_i t_i} \end{bmatrix}. \tag{45}$$

The boundary conditions at layer interfaces must be modified to account for the anisotropic case and therefore yield a new definition of the reflection ratio and the boundary condition matrix elements $\widetilde{X}_i^{\Pi}$ and $\widetilde{Y}_i^{\Pi}$.

$$\widetilde{R}_i^{\Pi} = \left[\frac{\widetilde{X}_i^{\Pi}(\widetilde{R}_{i+1}^{\Pi} + 1) + \widetilde{Y}_i^{\Pi}(\widetilde{R}_{i+1}^{\Pi} - 1)}{\widetilde{X}_i^{\Pi}(\widetilde{R}_{i+1}^{\Pi} + 1) - \widetilde{Y}_i^{\Pi}(\widetilde{R}_{i+1}^{\Pi} - 1)} \right] e^{-2v_i t_i} \tag{46}$$

where

$$\widetilde{X}_i^{\Pi} = \frac{v_i}{v_{i+1}} \frac{K_{i+1}^2}{K_i^2}, \quad \widetilde{Y}_i^{\Pi} = \frac{\alpha_{v,i}}{\alpha_{v,i+1}}.$$

Thus the changes introduced by the anisotropic model alter the continuation operator and boundary conditions of the TM modal potential. The other change enters in the actual differential equations governing the modal potentials. Notice that all expressions presented in Table 1 reduce to the isotropic case when $\alpha_v = \alpha_h = \alpha$ so that $K^2 = 1$ and $u = v$. Also, one should expect that fields generated by horizontal circulations of current (PM mode) should be unaffected by transverse isotropy, which is indeed the case.

Table 1a

$$\widetilde{\underset{\sim}{\mathcal{F}}}_e = dl \begin{bmatrix} AI'_{x,e} & AI'_{y,e} & CI'_{z,e} & 0 & -i\omega\mu_R \dfrac{\widetilde{T}_1}{u_S} I'_{x,e} \\ -AI'_{y,e} & AI'_{x,e} & 0 & CI'_{z,e} & -\dfrac{v_R}{K_R}\dfrac{\widetilde{P}_3}{\alpha_{v,S}} I'_{y,e} \\ 0 & 0 & EI'_{x,e} & EI'_{y,e} & -\lambda^2 K_S \dfrac{\widetilde{P}_1}{\alpha_{v,S} v_S} I'_{z,e} \\ BI'_{y,e} & -BI'_{x,e} & 0 & DI'_{z,e} & \mu_R \dfrac{\alpha_{v,R}}{\alpha_{v,S}} \widetilde{P}_4 I'_{y,e} \\ BI'_{x,e} & BI'_{y,e} & -DI'_{z,e} & 0 & -\mu_R \dfrac{u_R}{u_S} \widetilde{T}_2 I'_{x,e} \\ 0 & 0 & -FI'_{y,e} & FI'_{x,e} & 0 \end{bmatrix}$$

where

$$A = \left(\frac{v_R}{K_R} \frac{\widetilde{P}_3}{\alpha_{v,S}} - i\omega\mu_R \frac{\widetilde{T}_1}{u_S} \right), \qquad B = \left(\mu_R \frac{\alpha_{v,R}}{\alpha_{v,S}} \widetilde{P}_4 - \mu_R \frac{u_R}{u_S} \widetilde{T}_2 \right),$$

$$C = \frac{v_R}{v_S} \frac{K_S}{K_R} \frac{\widetilde{P}_2}{\alpha_{v,S}}, \qquad D = \mu_R \frac{\alpha_{v,R}}{\alpha_{v,S}} \frac{K_S \widetilde{P}_1}{v_S},$$

and

$$E = \frac{\widetilde{P}_4}{\alpha_{v,S}}, \qquad F = \mu_R \frac{\widetilde{T}_1}{u_S}.$$

TABLE 1b

$$\widetilde{\underset{\sim}{\mathcal{F}}}_m = i\omega\mu_S dA \begin{bmatrix} -AI'_{y,m} & AI'_{x,m} & 0 & CI'_{z,m} & -\dfrac{\mu_R}{\mu_S}\widetilde{T}_4 I'_{y,m} \\ -AI'_{x,m} & -AI'_{y,m} & -CI'_{z,m} & 0 & \dfrac{K_S}{K_R}\dfrac{v_R}{v_S}\widetilde{P}_2 I'_{x,m} \\ 0 & 0 & -EI'_{y,m} & -EI'_{x,m} & 0 \\ BI'_{x,m} & BI'_{y,m} & DI'_{z,m} & 0 & \mu_R\alpha_{v,R}\dfrac{K_S\widetilde{P}_1}{v_S} I'_{x,m} \\ -BI'_{y,m} & BI'_{x,m} & 0 & DI'_{z,m} & -\mu_R\dfrac{u_R\widetilde{T}_3}{\beta_S} I'_{y,m} \\ 0 & 0 & FI'_{x,m} & FI'_{y,m} & -\mu_R\lambda^2\dfrac{\widetilde{T}_1}{u_S\beta_S} I'_{z,m} \end{bmatrix}$$

where

$$A = \left(\frac{\mu_R}{\mu_S}\widetilde{T}_4 - \frac{K_S}{K_R}\frac{v_R}{v_S}\widetilde{P}_2\right). \qquad B = \mu_R\left(\frac{u_R\widetilde{T}_3}{\beta_S} - \alpha_{v,R}\frac{K_S\widetilde{P}_1}{v_S}\right),$$

$$C = \frac{\mu_R}{\mu_S}\frac{\widetilde{T}_1}{u_S}, \qquad D = \mu_R\frac{u_R}{u_S}\frac{\widetilde{T}_2}{\beta_S},$$

and

$$E = \frac{K_S\widetilde{P}_1}{v_S}, \qquad F = \mu_R\frac{\widetilde{T}_4}{\beta_S}.$$

Special Cases

One useful generalization of the above derivation is to find the electric potential of a pole source (i.e., the DC resistivity case). The potential can be obtained by integrating the electric field along the source and over the receiver dipole. For example, it is easy to show from equation (30) that the electric potential generated by a pole source (i.e. after integrating the source term from 0 to ∞) is

$$V_{pole} = \frac{Idl}{4\pi}\int_0^\infty \left(\frac{v_R}{K_R}\frac{\widetilde{P}_3}{\alpha_{v,S}} - i\omega\mu_R\frac{\widetilde{T}_1}{u_S}\right)\frac{J_0(\lambda\xi)}{\lambda}d\lambda. \tag{47}$$

For an isotropic earth at zero frequency, this term reduces to

$$V_{pole} = \frac{Idl}{4\pi}\int_0^\infty \left(\frac{\widetilde{P}_3}{\sigma_{v,S}}\right)J_0(\lambda\xi)d\lambda \tag{48}$$

Based on the above derivation of P_3 for a uniform halfspace, direct substitution yields

$$V_{pole} = \frac{Idl}{2\pi\sigma r}. \tag{49}$$

Thus equation (48) can be used to find the electrical response of any electrode array by superposition.

Another relevant generalization is to consider a finite source having arbitrary topology. That is, the source could be an extended bipole that follows a curvilinear path, but which is grounded at both ends. Alternatively, the source could be a polygonal loop of wire that closes on itself, i.e. is not grounded at any point. Such arbitrary sources can always be constructed from integrating the fields of an electric dipole along the path of the wire (since the electric fields are conservative). However, care must be taken in the integration since the terms associated with the double spatial derivatives of the EM kernel functions (see equation (30)) must be evaluated only at the grounding points. To be more specific, we rewrite equation (30) to indicate the integration over the source and specifically isolate the path dependent terms

$$\begin{bmatrix} E_x \\ E_y \\ E_z \\ B_x \\ B_y \\ B_z \end{bmatrix} = \frac{I}{4\pi}\int_0^\infty \underset{\sim}{\widetilde{\mathcal{F}}} \begin{bmatrix} \lambda^{-2}\partial_{xx} \\ \lambda^{-2}\partial_{xy} \\ 0 \\ 0 \\ 0 \end{bmatrix} \lambda J_0(\lambda\xi)\,d\lambda + \frac{I}{4\pi}\int_0^\infty \underset{\sim}{\widetilde{\mathcal{F}}} \begin{bmatrix} 0 \\ 0 \\ \partial_x \\ \partial_y \\ 1 \end{bmatrix} \lambda J_0(\lambda\xi)\,d\lambda \tag{50}$$

The first integral is ignored when the source closes upon itself (i.e., is a loop). If the wire bends between the grounding points, the first contribution of the first integral is only dependent on the location of the grounding points, while the second integral is evaluated over the path taken by the wire.

3 Routines

The algorithm outlined above has been coded into a general purpose FORTRAN routine that is able to compute the fields from any orientation of source and receiver, in a completely arbitrary earth. The numerical evaluation of the Hankel transforms is done using the FHT method, or direct integration, and is thus limited to the diffusive realm. No attempt has been made to optimize the code and there are many means of doing this. The primary purpose of this code is simply to illustrate the somewhat abstract theory described above in an algorithmic form.

The following provides listings of the various software routines used in the package.

The main program is `emdipole` and it calls the various input/output subroutines to define the layered earth model. For each frequency, `emdipole` computes all six electromagnetic field components using the prescribed source/receiver geometry. At the final step, the computed fields in the space domain are written to output files.

Listing of `emdipole.f`

```
1   c EMDIPOLE drives the subroutine package that calculates the
2   c electric and magnetic fields in a transversely isotropic layered
3   c earth.
4   c x - north  \
5   c y - east    > Right hand coordinate system.
6   c z - down   /
7   c
8         include  'dipole.inc'
9         integer  maxfre
10        parameter(maxfre=250)
11  c
12        real*8     sigmah(maxlay), sigmav(maxlay), epsiln(maxlay)
13        real*8     freq1, den, theta, radii(maxrad), freqmx
14        real*8     tol, freq(maxfre), tinc
15        complex*16 field(maxrad,6,maxfre)
16        integer    numrad, ndec, i, nfreq, ifre, irad, n1, n2
17        integer    fht_type, ntheta
18        character  fc(6)*2, aa*2, ab*2
19        data       fc/'Ex','Ey','Ez','Hx','Hy','Hz'/
20  c
21        call reader(maxlay,layers,sigmah,sigmav,epsiln,mu,
22       >            thckns,freq1,ndec,den,maxrad,numrad,radii,ntheta,
23       >            theta,tinc,js,ms,sz,rz,tol,fht_type)
24  c
25        nfreq = nint(den*ndec)
26        freqmx=freq1*(10.0**float(ndec))
27        call look(maxlay,layers,sigmah,sigmav,epsiln,mu,
28       >          thckns,freq1,freqmx,den,maxrad,numrad,radii,
29       >          ntheta,theta,tinc,js,ms,sz,rz)
30
31        call orient(wc,theta)
32        call initd
33        do i = 1, nfreq
34          freq(i) = freq1*10**((i-1)/den)
35          if(mod(i-1,den).eq.0) write(6,*) 'Frequency = ',freq(i)
36          call dipole(sigmah,sigmav,epsiln,freq(i),numrad,
37       >              radii(1),theta,tol,radii,field(1,1,i),
38       >              fht_type)
39        end do
40  c
41  c output
42  c
```

```
43        open(unit=11,file='ex.fld',status='unknown')
44        open(unit=12,file='ey.fld',status='unknown')
45        open(unit=13,file='ez.fld',status='unknown')
46        open(unit=14,file='hx.fld',status='unknown')
47        open(unit=15,file='hy.fld',status='unknown')
48        open(unit=16,file='hz.fld',status='unknown')
49        open(unit=16,file='hz.fld',status='unknown')
50        do i = 1, 6
51          write(i+10,'(a)') '@xaxis  label "Frequency (Hz)"'
52          write(i+10,'(a)') '@xaxis  ticklabel format power'
53          write(i+10,'(a)') '@yaxis  label "'//fc(i)//' Amplitude"'
54        end do
55        do irad = 1, numrad
56          call itoa(aa,2*irad-2,n1)
57          call itoa(ab,2*irad-1,n2)
58          do i = 1, 6
59            write(i+10,'(a)') '@s'//aa(1:n1)//' color 2'
60            write(i+10,'(a)') '@s'//ab(1:n2)//' color 1'
61            write(i+10,'(a1,2x,f15.6)') '# ',radii(irad)
62            do ifre = 1, nfreq
63              write(i+10,*) freq(ifre),
64       >                        (dreal(field(irad,i,ifre))),
65       >                        (dimag(field(irad,i,ifre)))
66            end do
67            write(i+10,*) '>'
68          end do
69        end do
70        close(unit=11)
71        close(unit=12)
72        close(unit=13)
73        close(unit=14)
74        close(unit=15)
75        close(unit=16)
76
77        stop
78  1     format(13(1pe14.7,1x))
79        end
```

The following file is included in almost all other routines and provides the common parameter definitions and array dimensions.

Listing of `dipole.inc`

```
1   c
2   c sz     = depth of the source
3   c s      = layer containing the source
4   c sdepup = distance to nearest interface above the source
5   c sdepdn = distance to nearest interface below the source
6   c sdist  = distance to interface nearest the source, but between
7   c          the source and receiver
8   c
9   c rz     = depth of the receiver
10  c r      = layer containing the receiver
11  c rdepup = distance to nearest interface above the receiver
12  c rdepdn = distance to nearest interface below the receiver
13  c rdist  = distance to interface nearest the receiver, but between
14  c          the source and receiver
15  c
16  c alphah = horizontal admittivity vector
17  c alphav = vertical admittivity vector
18  c beta   = impedivity vector
19  c kappa  = transverse isotropy vector
20  c
21        integer    maxlay, maxrad
22        parameter (maxlay = 100, maxrad = 50)
```

```
c
      complex*16 alphah(maxlay), alphav(maxlay)
      complex*16 beta(maxlay),   kappa(maxlay)
      real*8     mu(maxlay), pi/3.14159265358979/
      real*8     thckns(maxlay), rdist, sdist, rz, sz
      real*8     sdepup, sdepdn, wc(2,5), js(3), ms(3)
      integer    r, s, layers
c
      common /properties/ alphah, alphav, beta, kappa, mu
      common /structure/  thckns, layers
      common /source/     sdist, sdepup, sdepdn, sz, js, ms, s
      common /receiver/   rdist, rz, r
      common /geometry/   wc
```

`dipole` computes the EM fields for the input geometry and layered earth model at a single frequency. This routine essentially computes equation (32).

Listing of `dipole.f`

```
c-----------------------------------------------------------------------
      subroutine dipole(sigmah,sigmav,epsiln,freq,numrad,
     >                  r1,theta,tol,radii,tfield,fht_type)
c-----------------------------------------------------------dipole
c computes the EM fields due to any electric or magnetic dipole source
c within any arbitrarily layered anisotropic earth.
c the routine works in the frequency domain using lagged convolution to
c generate the fields at various radii, along a single angle.
c
c input variables:
c         sigmah ---- a vector containing the horizontal conductivities of
c                     each layer. the air halfspace is a layer.
c         sigmav ---- a vector containing the vertical conductivity of
c                       each layer.
c         epsiln ---- a vector containing the values of epsilon_0 for each
c                     layer.
c         freq ------ frequency (in hz) for which the response is
c                       calculated.
c         numrad ---- number of radii at which to calculate the fields.
c         r1 -------- initial radius value
c         theta ----- the angle which the receiver makes with the source.
c       tol   ----- FHT tolerance
c       fht_type -- select FHT method or Pade approximate method
c
c common variables:
c
c         layers ---- the number of layers in the model.
c         thckns ---- model thickness vector
c         sz -------- z coordinate of the transmitter (z positive down)
c         rz -------- z coordinate of the receiver (z positive down)
c         js(3) ----- electric source vector
c         ms(3) ----- magnetic source vector
c
c output variables:
c         radii ----- a vector of exponentially spaced radius values.
c                     the first element is r1, and the vector is filled
c                     by the routine.
c         tfield ---- a vector of the field values as a function radius
c
      external response
c
      include 'dipole.inc'
      real*8  delta, tol, height
c
      common /density/ delta

      integer    i, j, numrad, nr, fht_type
```

```
      real*8     radii(maxrad), theta, freq, r1
      real*8     sigmah(maxlay), sigmav(maxlay), epsiln(maxlay)
      complex*16 tfield(maxrad,6), rfield(maxrad,6)
      logical    newh
c
      nr = numrad
      do i = 1, nr
        radii(i) = r1*exp(delta*(i-1))
      end do
      height = abs(sz-rz)
c
c set up the frequency dependent variables
c
      call initw(sigmah,sigmav,epsiln,freq)
c
c Begin the hankel transform routines
c
      newh = .true.
c
c J0
c
      do i = 1, 6
        if(fht_type.eq.1) then
          call fht(newh,nr,radii,height,0,tol,response,i,tfield(1,i))
         else
          do j = 1, numrad
            call hantrn(tfield(j,i),0,radii(j),i,tol)
          end do
        end if
      end do
c
c J1 terms with a 1/r dependance
c
      do i = 1, 6
        if(fht_type.eq.1) then
          call fht(newh,nr,radii,height,1,tol,response,i+6,rfield(1,i))
         else
          do j = 1, numrad
            call hantrn(rfield(j,i),1,radii(j),i+6,tol)
          end do
        end if
c
        do j = 1, numrad
          tfield(j,i) = tfield(j,i) + rfield(j,i)/radii(j)
        end do
      end do
c
c J1 terms without the 1/r dependance
c
      do i = 1, 6
        if(fht_type.eq.1) then
          call fht(newh,nr,radii,height,1,tol,response,i+12,rfield(1,i))
         else
          do j = 1, numrad
            call hantrn(rfield(j,i),1,radii(j),i+12,tol)
          end do
        end if
c
        do j = 1, numrad
          tfield(j,i) = tfield(j,i) + rfield(j,i)
        end do
      end do
      return
      end
c-----------------------------------------------------------------------
      subroutine hantrn(field,order,rad,n,tol)
c-----------------------------------------------------------------------
      external   funct
      integer    order, nker, ierr, n, ic, ij, ir
      complex*16 field
      real*8     tol, rad
      logical    new/.true./
      character*2 ls(6)/'Ex','Ey','Ez','Hx','Hy','Hz'/
```

```
121         character*2 bs(2)/'J0','J1'/, rs(2)/'/r',' '/
122         common     /fun/ nker
123   c
124         nker = n
125   c      call besautz(field,order,1,7,rad,funct,tol,0.1*tol,1,new,ierr)
126         write(6,*) 'BESAUTZ - routine missing !'
127         stop
128
129         if(ierr.eq.1) then
130           if(n.le.6) then
131             ic = n
132             ij = 1
133             ir = 2
134            else if (n.gt.6.and.n.le.12) then
135             ic = n-6
136             ij = 2
137             ir = 1
138            else
139             ic = n-12
140             ij = 2
141             ir = 2
142           end if
143           write(6,*) 'BESAUT failed to converge; ',ls(ic),
144        >             ' ',bs(ij),rs(ir)
145         end if
146         return
147         end
148   c-------------------------------------------------------------------------
149         complex*16 function funct(lambda)
150   c-------------------------------------------------------------------------
151         external   response
152         real*8     lambda
153         integer    iker, nker
154         complex*16 kern(-45:301,18), temp, response
155   c
156         common /kernels/ kern, iker
157         common     /fun/ nker
158   c
159         iker = 1
160         temp = response(lambda)
161         funct = kern(iker,nker)
162   c
163         return
164         end
```

dipole provides the option of not using the FHT algorithm and computing the Hankel transform by direct integration (Chave 1983). This approach is generally more accurate than the FHT convolution method, but is also much more time consuming. This routine requires the subroutine besautz and associated subroutines available from A.D. Chave. For this release, a besautz stub routine is provided, but it just returns an error code. To exploit this option, obtain besautz, compile and link it with the emdipole code.

reflctn computes the reflection coefficients at the source level, both above and below the source. The subroutine uses recurrsion based on an initial value of $R_N = 0$ and equations (22) and (46).

Listing of reflctn.f

```
1   c-------------------------------------------------------------------
2         subroutine reflctn(u,v,rtor,rpol,lamsq,rs_pol,rs_tor)
3   c-------------------------------------------------------------------
4   c calculate the reflection coefficents for the source
5   c level looking both ways in the stack from the source level.
6   c
7         include 'dipole.inc'
8         integer  i
```

```
      complex*16 u(maxlay),v(maxlay),rtor(maxlay),rpol(maxlay)
      complex*16 gamma2,a,b,expont,temp,rs_pol,rs_tor
      real*8     lamsq,distan
c
c initialize the lambda dependent variables.
c
      do i = 1, layers
        gamma2 = alphah(i)*beta(i)
        u(i)   = cdsqrt(         lamsq + gamma2)
        v(i)   = cdsqrt(kappa(i)*lamsq + gamma2)
      end do
c
c find the reflection coefficent above the source.
c
      rpol(1) = dcmplx(0.0d0,0.0d0)
      rtor(1) = dcmplx(0.0d0,0.0d0)
c
      do i = 2, s
        if(i.eq.s) then
          distan = -2.0*sdepup
         else
          distan = -2.0*thckns(i)
        end if
c
        expont = cdexp(v(i)*distan)
c
        a = v(i)*alphah(i-1) + v(i-1)*alphah(i)
        b = v(i)*alphah(i-1) - v(i-1)*alphah(i)
        rpol(i) = (rpol(i-1)*a + b)*expont/(rpol(i-1)*b + a)
c
        expont = cdexp(u(i)*distan)
        a = u(i)*beta(i-1) + u(i-1)*beta(i)
        b = u(i)*beta(i-1) - u(i-1)*beta(i)
        rtor(i) = (rtor(i-1)*a + b)*expont/(rtor(i-1)*b + a)
      end do
      rs_pol = rpol(s)
      rs_tor = rtor(s)
c
c Find the reflection coefficent below the source.
c
      rpol(layers) = dcmplx(0.0d0,0.0d0)
      rtor(layers) = dcmplx(0.0d0,0.0d0)
c
      do i = layers-1, s, -1
        if(i.eq.s) then
          distan = -2.0*sdepdn
         else
          distan = -2.0*thckns(i)
        end if
c
        expont = cdexp(v(i)*distan)
        a = v(i)*alphah(i+1) + v(i+1)*alphah(i)
        b = v(i)*alphah(i+1) - v(i+1)*alphah(i)
        rpol(i) = (rpol(i+1)*a + b)*expont/(rpol(i+1)*b + a)
c
        expont = cdexp(u(i)*distan)
        a = u(i)*beta(i+1) + u(i+1)*beta(i)
        b = u(i)*beta(i+1) - u(i+1)*beta(i)
        rtor(i) = (rtor(i+1)*a + b)*expont/(rtor(i+1)*b + a)
      end do
c
c The ordering of the reflection coefficents is important. rs_pol
c and rs_tor should be the reflection coefficients on the
c side of the source away from the receiver.
c
      if(sz.gt.rz) then
        temp    =  rpol(s)
        rpol(s) =  rs_pol
        rs_pol  =  temp
```

```
c
          temp    =  rtor(s)
          rtor(s) =  rs_tor
          rs_tor  =  temp
        end if
        return
        end
```

proptn computes the propagation matrices that use the reflection coefficients to take the primary potentials at the source level to the receiver level. The relevant equations are (19) and (20), as well as (24) and (25).

Listing of proptn.f

```
c-----------------------------------------------------------------------
      subroutine proptn(u,v,rtor,rpol,rs_pol,rs_tor,t,p)
c-----------------------------------------------------------------------prop
c generate the propagation matrices fromc the source to the receiver
c
      include 'dipole.inc'
c
      integer    i, step
      real*8     distan
      complex*16 proppm(2,2), proptm(2,2), pmprop(2,2), tmprop(2,2)
      complex*16 contpm(2,2), conttm(2,2), fact, rs_pol, rs_tor
      complex*16 u(maxlay), v(maxlay), rpol(maxlay), rtor(maxlay)
      complex*16 p(4),t(4)
c
c calculate the total potentials at the source level.
c - note the extra factor for this matrix added to the summation
c   definitions below
c
      pmprop(1,1) = 1.0d0
      pmprop(1,2) = rs_pol
      pmprop(2,1) = rpol(s)
      pmprop(2,2) = rpol(s)*rs_pol
c
      tmprop(1,1) = 1.0d0
      tmprop(1,2) = rs_tor
      tmprop(2,1) = rtor(s)
      tmprop(2,2) = rtor(s)*rs_tor
c
c continue the source level total potentials to the nearest interface
c
      step = 1
      if(r.lt.s) step = -1
      do i = s, r-step, step
c
        distan = thckns(i)
        if(i.eq.s) distan = sdist
        call cont(contpm,conttm,u(i),v(i),distan)
c
        call matmult(contpm,pmprop,pmprop)
        call matmult(conttm,tmprop,tmprop)
c
c the potential that is propagating away from the source level is then
c continued across the interface, and the potential propagating towards
c the source is found from the potential propagating away from the source.
c this is done using the reflection coefficents to help eliminate
c round-off errors.
c
        fact = 0.5 /(alphav(i+step)*kappa(i)*v(i+step))
        proppm(1,1) = fact*(alphah(i)*v(i+step) + alphah(i+step)*v(i))
        proppm(1,2) = fact*(alphah(i)*v(i+step) - alphah(i+step)*v(i))
        proppm(2,1) = proppm(1,1)*rpol(i+step)
```

```
52          proppm(2,2) = proppm(1,2)*rpol(i+step)
53  c
54          fact = 0.5 /(u(i+step)*beta(i+step))
55          proptm(1,1) = fact*(beta(i)*u(i+step) + beta(i+step)*u(i))
56          proptm(1,2) = fact*(beta(i)*u(i+step) - beta(i+step)*u(i))
57          proptm(2,1) = proptm(1,1)*rtor(i+step)
58          proptm(2,2) = proptm(1,2)*rtor(i+step)
59  c
60  c the contribution from this interface is multiplied into the
61  c propagation matrix.
62  c
63          call matmult(proppm,pmprop,pmprop)
64          call matmult(proptm,tmprop,tmprop)
65        end do
66  c
67  c once the receiver layer is reached, we must propagate from the
68  c boundary to the receiver level
69  c
70        call cont(contpm,conttm,u(r),v(r),rdist)
71        call matmult(contpm,pmprop,pmprop)
72        call matmult(conttm,tmprop,tmprop)
73  c
74        fact = 1.0d0 / ( 1.0d0 - rpol(s)*rs_pol )
75        p(1)=(pmprop(1,1)+pmprop(1,2)+(pmprop(2,1)+pmprop(2,2)))*fact
76        p(2)=(pmprop(1,1)+pmprop(1,2)-(pmprop(2,1)+pmprop(2,2)))*fact
77        p(3)=(pmprop(1,1)-pmprop(1,2)-(pmprop(2,1)-pmprop(2,2)))*fact
78        p(4)=(pmprop(1,1)-pmprop(1,2)+(pmprop(2,1)-pmprop(2,2)))*fact
79  c
80        fact = 1.0d0 / ( 1.0d0 - rtor(s)*rs_tor )
81        t(1)=(tmprop(1,1)+tmprop(1,2)+(tmprop(2,1)+tmprop(2,2)))*fact
82        t(2)=(tmprop(1,1)+tmprop(1,2)-(tmprop(2,1)+tmprop(2,2)))*fact
83        t(3)=(tmprop(1,1)-tmprop(1,2)-(tmprop(2,1)-tmprop(2,2)))*fact
84        t(4)=(tmprop(1,1)-tmprop(1,2)+(tmprop(2,1)-tmprop(2,2)))*fact
85  c
86        return
87        end
```

cont is the routine to continue the potentials vertically across a uniform media. The equations used are (23) and (45). As the two non-zero elements of this matrix are inverses of each other, only the term with the negative expoential is calculated and inverted. This saves expensive complex exponential evaluations.

Listing of cont.f

```
1   c-----------------------------------------------------------------------
2         subroutine cont(contpm,conttm,u,v,distan)
3   c-----------------------------------------------------------------------cont
4   c cont generates an continuation matrix for pi and gamma. u and v are
5   c the attenuation factors for the layer to be continued across, and
6   c distan is the distance through the layer.
7   c
8         complex*16 contpm(2,2), conttm(2,2), u, v
9         real*8     distan
10  c
11        if(abs(u*distan).gt.80) then
12          conttm(1,1) = dcmplx(0.0d0,0.0d0)
13          conttm(2,2) = dcmplx(0.0d0,0.0d0)
14         else
15          conttm(1,1) = cdexp(-u*distan)
16          conttm(2,2) = 1.0d0/conttm(1,1)
17        end if
18  c
19        conttm(1,2) = dcmplx(0.0d0,0.0d0)
20        conttm(2,1) = dcmplx(0.0d0,0.0d0)
```

```
c
      if(abs(v*distan).gt.80) then
        contpm(1,1) = dcmplx(0.0d0,0.0d0)
        contpm(2,2) = dcmplx(0.0d0,0.0d0)
       else
        contpm(1,1) = cdexp(-v*distan)
        contpm(2,2) = 1.0d0/contpm(1,1)
      end if
c
      contpm(1,2) = dcmplx(0.0d0,0.0d0)
      contpm(2,1) = dcmplx(0.0d0,0.0d0)
      return
      end
```

response clculates the wavenumber domain response of the layered earth to a single wavenumber λ of source excitation. This routine is called as a function by the Fast Hankel Transform routine fht.

Listing of response.f

```
c-----------------------------------------------------------------------
      complex function response(lambda)
c-----------------------------------------------------------------------
c Calculates the response in the Hankel domain of the
c layered earth to a given dipole excitation.
c
      include  'dipole.inc'
c
      integer    iker, i, j
      complex*16 u(maxlay), v(maxlay), rs_pol, rs_tor
      complex*16 rpol(maxlay), rtor(maxlay), p(4), t(4)
      complex*16 hfield(6,5), kern(-45:301,18)
      real*8     lambda, lamsq
c
      common /kernels/ kern, iker
c
      lamsq = lambda*lambda
c
c compute the reflection coefficients
c
      call reflctn(u,v,rtor,rpol,lamsq,rs_pol,rs_tor)
c
c compute the propagation matrices
c
      call proptn (u,v,rtor,rpol,rs_pol,rs_tor,t,p)
c
c introduce the electric dipole source normalization
c
      if(js(1).ne.0.or.js(2).ne.0.or.js(3).ne.0) then
        call jsrch(u,v,lamsq,t,p,hfield)
c
c introduce the magnetic dipole source normalization
c
       else if(ms(1).ne.0.or.ms(2).ne.0.or.ms(3).ne.0) then
        call msrch(u,v,lamsq,t,p,hfield)
      end if
c
c J0 kernels
c
      do j = 1, 6
        kern(iker,j) = cmplx(0.0,0.0)
        do i = 1, 5
          kern(iker,j) = kern(iker,j) + wc(1,i)*hfield(j,i)*lambda
        end do
      end do
c
c J1 kernels with an r dependance
```

```
c
      do j = 1, 6
        kern(iker,j+6) = cmplx(0.0,0.0)
        do i = 1, 2
          kern(iker,j+6) = kern(iker,j+6) + wc(2,i)*hfield(j,i)
        end do
      end do
c
c J1 kernels without an r dependance
c
      do j = 1, 6
        kern(iker,j+12) = cmplx(0.0,0.0)
        do i = 3, 5
          kern(iker,j+12) = kern(iker,j+12) +
     >                      wc(2,i)*hfield(j,i)*lambda**2
        end do
      end do
c
      return
      end
```

hantrn provides the option of not using the FHT algorithm and computing the Hankel transform by direct integration (Chave 1983). This approach is generally more accurate than the FHT convolution method, but is also much more time consuming. This routine requires the subroutine besautz and associated subroutines available from A.D. Chave. For this release, a besautz stub routine is providedi, but it just returns an error code. To exploit this option, obtain besautz, compile and link it with the emdipole code.

fht is the primary computational routine. It calculates the lagged convolution of the layered earth kernel function (computed in response) with the Fast Hankel Transform filter coefficients. All kernels are calculated for each wavenumber, but the FHT convergence checking is done sequentially for each kernel to ensure accuracy. The unused kernels are saved for later evaluation.

Listing of fht.f

```
c-----------------------------------------------------------------------fht
      subroutine fht(new,nrad,radii,hgt,order,tol,kernel,nker,field)
c-----------------------------------------------------------------------
c this subroutine computes the zeroth or first order hankel transform
c for any diffusive kernel function using lagged convolution.
c
c new........is a logical variable set to .true. each time a new lagged
c            convolution is performed (ie for each new frequency).
c
c nrad.......is the number radii at which the field is to be calculated.
c            when using lagged convolution, it is advantageous to evaluate
c            the field at many radial points.
c
c radii......is a vector of length nrad containing the radii at
c            which the field is to be evaluated. the user must generate
c            this array from delta in the common block /density/ using
c            the expression;
c
c                    radii(i) = radii(1)*exp(delta*(i-1))    i=1,nrad
c
c            where radii(1) is the minimum radius. delta is the sampling
c            density of the filter weights for fht.
c
c hgt........is the vertical separation between the source and receiver
c
c order......is the order of the hankel transform (0 or 1).
c
c tol........is the desired tolerence of the field caculated. essentially
```

```
29  c             determines the number of significant figures in the final
30  c             answer. tests for convergence are made by comparing the
31  c             magnitude of the next term in the sum to tol*(summed total).
32  c             if the next term in the sum is smaller than this value, the
33  c             series is assumed to have converged.
34  c
35  c kernel.....is a user defined complex function of lambda, ie.
36  c             complex function kernel(lambda). the results of the
37  c             evaluation of the kernel function are stored in the
38  c             common block /kernels/ in the array kern. the variable
39  c               iker points to the storage location for a given wavenumber
40  c               (lambda), and the variable nker points to the hankel
41  c             transform kernel. allows the user to store several kernel
42  c             functions that can be computed simultaneously and then
43  c             Hankel transform each kernel. the array kern must be
44  c             dimensioned to match the number of kernels to be evaluated.
45  c
46  c nker.......is the array indicator for the kernel function. ie determines
47  c             which kernel is to be transformed.
48  c
49  c field......are the fields variations with radius for each of the kernel
50  c             vectors and is dimensioned fld(nrad)
51  c
52        external kernel
53  c
54  c this include statement inputs the filter weights
55  c
56        include 'fhtwts.inc'
57  c
58        integer    irad, iker, ifilt, nker, nrad, order
59        integer    istep, kstart
60        complex*16 field(nrad), f, temp, kern(-45:301,18), kernel
61        real*8     radii(nrad), tr, ti, tmaxr, tmaxi, tol, coef
62        real*8     hgt, rho, error1, error2, lambda(-45:301)
63        logical    converged, new, stored(-45:301)
64  c
65  c user supplied common block /kernels/.
66  c
67        common /kernels/ kern, iker
68        common /density/ delta
69  c
70  c initialize the lambda vector
71  c
72        if(new) then
73          do irad = -nrad, nfilt
74            lambda(irad) = lambda1*exp(delta*(irad-1))/radii(1)
75          end do
76  c
77  c find the starting position in lambda space which is near the peak of
78  c the kernel function. it appears when rho = 1, this peak is at 1.0e-03.
79  c
80          rho    = sqrt( radii(1)*radii(1) + hgt*hgt )
81          kstart = dlog(1.0e-03/(lambda1*rho))/delta
82  c
83  c initialize a vector which keeps track of the kernel functions computed.
84  c
85          do irad = -nrad, nfilt
86            stored(irad) = .false.
87          end do
88          new = .false.
89        end if
90  c
91  c evaluate the transformation over the radii values.
92  c
93        do irad = 1, nrad
94          field(irad) = cmplx(0.0,0.0)
95  c
96  c hankel transform loop over lambda
97  c
98          converged = .false.
99          istep  = 1
100         iker   = kstart
101         ifilt  = kstart + irad - 1
```

```
102           do while(.not.converged)
103   c
104   c if the kernel functions have not been evaluated at the required value of
105   c lambda, then they must be found and stored in the common block kernels.
106   c
107             if(.not.stored(iker)) then
108               temp = kernel(lambda(iker))
109               stored(iker) = .true.
110             end if
111   c
112   c evaluate next term in the summation in real and imaginary
113   c parts to improve the computational speed.
114   c
115             if(order.eq.0) then
116               coef = dble(wt0(ifilt))
117              else if(order.eq.1) then
118               coef = dble(wt1(ifilt))
119             end if
120             f = coef*kern(iker,nker)
121             field(irad) = field(irad) + f
122   c
123   c evaluate 20 terms in the series before testing for convergence.
124   c
125             if(iker.gt.kstart+20.or.istep.lt.0) then
126               tr    = dabs(dreal(f))
127               tmaxr = dabs(dreal(field(irad)))*tol
128               ti    = dabs(dimag(f))
129               tmaxi = dabs(dimag(field(irad)))*tol
130   c
131   c test for convergence
132   c
133               if(tr.le.tmaxr.and.ti.le.tmaxi) then
134   c
135   c if converged on the rhs, start summation on lhs.
136   c
137                 if(istep.gt.0) then
138                   iker = kstart
139                   istep = -1
140                  else if(istep.lt.0) then
141   c
142   c if converged on both sides, return normalized field value.
143   c
144                   field(irad) = field(irad) / radii(irad)
145                   converged = .true.
146                 end if
147               end if
148             end if
149   c
150   c increment the filter and kernel counters for next terms in series.
151   c
152             iker  = iker + istep
153             ifilt = iker + irad - 1
154   c
155   c check to see if filter limits have been reached.
156   c
157             if((ifilt.gt.nfilt.or.ifilt.lt.1).and..not.converged) then
158               error1 = 100.0*dreal(f)/dreal(field(irad))
159               error2 = 100.0*dimag(f)/dimag(field(irad))
160               if(ifilt.gt.nfilt) write(6,*)'+fht overflow: kernel ',nker
161               if(ifilt.lt.1) write(6,*)    '-fht overflow: kernel ',nker
162               write(6,*) '% error : real ',error1,'; imag ',error2
163   c
164   c continue the convolution if the lhs has not been done
165   c
166               if(ifilt.lt.1) then
167                 field(irad)=field(irad)/radii(irad)
168                 converged = .true.
169                else
170                 iker  = kstart - 1
171                 ifilt = iker + irad - 1
172                 istep = -1
173               end if
174             end if
```

```
175           end do
176         end do
177         return
178         end
```

The FHT filter coefficients were calculated by the method of Johansen and Sørensen (1979), as implemented by Christensen (1990).

Listing of `fhtwts.inc`

```
1   C ------
2         integer nfilt/301/
3         real    wt0(301), wt1(301)
4         real    lambda1/ 1.0000000E-20/
5         real*8  delta/ 2.3025849E-01/
6   C ----------
7         data wt0 /
8        + 2.3025128E-21, 2.8986919E-21, 3.6492369E-21, 4.5941171E-21,
9        + 5.7836507E-21, 7.2811849E-21, 9.1664687E-21, 1.1539900E-20,
10       + 1.4527874E-20, 1.8289510E-20, 2.3025128E-20, 2.8986919E-20,
11       + 3.6492369E-20, 4.5941171E-20, 5.7836507E-20, 7.2811849E-20,
12       + 9.1664687E-20, 1.1539900E-19, 1.4527874E-19, 1.8289510E-19,
13       + 2.3025128E-19, 2.8986919E-19, 3.6492369E-19, 4.5941171E-19,
14       + 5.7836507E-19, 7.2811849E-19, 9.1664687E-19, 1.1539900E-18,
15       + 1.4527874E-18, 1.8289510E-18, 2.3025128E-18, 2.8986919E-18,
16       + 3.6492369E-18, 4.5941171E-18, 5.7836507E-18, 7.2811849E-18,
17       + 9.1664687E-18, 1.1539900E-17, 1.4527874E-17, 1.8289510E-17,
18       + 2.3025128E-17, 2.8986919E-17, 3.6492369E-17, 4.5941171E-17,
19       + 5.7836507E-17, 7.2811849E-17, 9.1664687E-17, 1.1539900E-16,
20       + 1.4527874E-16, 1.8289510E-16, 2.3025128E-16, 2.8986919E-16,
21       + 3.6492369E-16, 4.5941171E-16, 5.7836507E-16, 7.2811849E-16,
22       + 9.1664687E-16, 1.1539900E-15, 1.4527874E-15, 1.8289510E-15,
23       + 2.3025128E-15, 2.8986919E-15, 3.6492369E-15, 4.5941171E-15,
24       + 5.7836507E-15, 7.2811849E-15, 9.1664687E-15, 1.1539900E-14,
25       + 1.4527874E-14, 1.8289510E-14, 2.3025128E-14, 2.8986919E-14,
26       + 3.6492369E-14, 4.5941171E-14, 5.7836507E-14, 7.2811849E-14,
27       + 9.1664687E-14, 1.1539900E-13, 1.4527874E-13, 1.8289510E-13,
28       + 2.3025128E-13, 2.8986919E-13, 3.6492369E-13, 4.5941171E-13,
29       + 5.7836507E-13, 7.2811849E-13, 9.1664687E-13, 1.1539900E-12,
30       + 1.4527874E-12, 1.8289510E-12, 2.3025128E-12, 2.8986919E-12,
31       + 3.6492369E-12, 4.5941171E-12, 5.7836507E-12, 7.2811849E-12,
32       + 9.1664687E-12, 1.1539900E-11, 1.4527874E-11, 1.8289510E-11,
33       + 2.3025128E-11, 2.8986919E-11, 3.6492369E-11, 4.5941171E-11,
34       + 5.7836507E-11, 7.2811849E-11, 9.1664687E-11, 1.1539900E-10,
35       + 1.4527874E-10, 1.8289510E-10, 2.3025128E-10, 2.8986919E-10,
36       + 3.6492369E-10, 4.5941171E-10, 5.7836507E-10, 7.2811849E-10,
37       + 9.1664687E-10, 1.1539900E-09, 1.4527874E-09, 1.8289510E-09,
38       + 2.3025128E-09, 2.8986919E-09, 3.6492369E-09, 4.5941171E-09,
39       + 5.7836507E-09, 7.2811849E-09, 9.1664687E-09, 1.1539900E-08,
40       + 1.4527874E-08, 1.8289510E-08, 2.3025128E-08, 2.8986919E-08,
41       + 3.6492369E-08, 4.5941171E-08, 5.7836507E-08, 7.2811849E-08,
42       + 9.1664687E-08, 1.1539900E-07, 1.4527874E-07, 1.8289510E-07,
43       + 2.3025128E-07, 2.8986919E-07, 3.6492369E-07, 4.5941171E-07,
44       + 5.7836507E-07, 7.2811849E-07, 9.1664687E-07, 1.1539900E-06,
45       + 1.4527874E-06, 1.8289510E-06, 2.3025128E-06, 2.8986919E-06,
46       + 3.6492369E-06, 4.5941171E-06, 5.7836507E-06, 7.2811849E-06,
47       + 9.1664687E-06, 1.1539900E-05, 1.4527874E-05, 1.8289509E-05,
48       + 2.3025128E-05, 2.8986919E-05, 3.6492369E-05, 4.5941170E-05,
49       + 5.7836506E-05, 7.2811847E-05, 9.1664683E-05, 1.1539900E-04,
50       + 1.4527872E-04, 1.8289507E-04, 2.3025123E-04, 2.8986908E-04,
51       + 3.6492346E-04, 4.5941125E-04, 5.7836416E-04, 7.2811667E-04,
52       + 9.1664324E-04, 1.1539828E-03, 1.4527729E-03, 1.8289221E-03,
53       + 2.3024553E-03, 2.8985771E-03, 3.6490077E-03, 4.5936598E-03,
54       + 5.7827384E-03, 7.2793646E-03, 9.1628369E-03, 1.1532654E-02,
55       + 1.4513417E-02, 1.8260669E-02, 2.2967599E-02, 2.8872171E-02,
56       + 3.6263558E-02, 4.5485028E-02, 5.6927766E-02, 7.1002701E-02,
57       + 8.8068521E-02, 1.0826428E-01, 1.3117586E-01, 1.5515822E-01,
58       + 1.7619271E-01, 1.8587700E-01, 1.6936511E-01, 1.0395977E-01,
```

```
     +-3.1093055E-02,-2.2662732E-01,-3.6303801E-01,-2.0549570E-01,
     + 3.4214651E-01, 3.0577295E-01,-5.0085931E-01, 3.0562527E-01,
     +-1.3553278E-01, 5.9188488E-02,-2.6407521E-02, 1.1037883E-02,
     +-4.4279470E-03, 1.7646411E-03,-7.0263272E-04, 2.7973045E-04,
     +-1.1136317E-04, 4.4334507E-05,-1.7649887E-05, 7.0265467E-06,
     +-2.7973186E-06, 1.1136326E-06,-4.4334513E-07, 1.7649887E-07,
     +-7.0265468E-08, 2.7973186E-08,-1.1136326E-08, 4.4334513E-09,
     +-1.7649887E-09, 7.0265468E-10,-2.7973186E-10, 1.1136326E-10,
     +-4.4334513E-11, 1.7649887E-11,-7.0265468E-12, 2.7973186E-12,
     +-1.1136326E-12, 4.4334513E-13,-1.7649887E-13, 7.0265468E-14,
     +-2.7973186E-14, 1.1136326E-14,-4.4334513E-15, 1.7649887E-15,
     +-7.0265468E-16, 2.7973186E-16,-1.1136326E-16, 4.4334513E-17,
     +-1.7649887E-17, 7.0265468E-18,-2.7973186E-18, 1.1136326E-18,
     +-4.4334513E-19, 1.7649887E-19,-7.0265468E-20, 2.7973186E-20,
     +-1.1136326E-20, 4.4334513E-21,-1.7649887E-21, 7.0265468E-22,
     +-2.7973186E-22, 1.1136326E-22,-4.4334513E-23, 1.7649887E-23,
     +-7.0265468E-24, 2.7973186E-24,-1.1136326E-24, 4.4334513E-25,
     +-1.7649887E-25, 7.0265468E-26,-2.7973186E-26, 1.1136326E-26,
     +-4.4334513E-27, 1.7649887E-27,-7.0265468E-28, 2.7973186E-28,
     +-1.1136326E-28, 4.4334513E-29,-1.7649887E-29, 7.0265468E-30,
     +-2.7973186E-30, 1.1136326E-30,-4.4334513E-31, 1.7649887E-31,
     +-7.0265468E-32, 2.7973186E-32,-1.1136326E-32, 4.4334513E-33,
     +-1.7649887E-33, 7.0265468E-34,-2.7973186E-34, 1.1136326E-34,
     +-4.4334513E-35, 1.7649887E-35,-7.0265468E-36, 2.7973186E-36,
     +-1.1136326E-36/

      data wt1 /
     + 1.1512925E-41 ,1.8246757E-41 ,2.8919159E-41 ,4.5833780E-41,
     + 7.2641643E-41 ,1.1512925E-40 ,1.8246756E-40 ,2.8919160E-40,
     + 4.5833779E-40 ,7.2641647E-40 ,1.1512925E-39 ,1.8246757E-39,
     + 2.8919159E-39 ,4.5833781E-39 ,7.2641647E-39 ,1.1512925E-38,
     + 1.8246757E-38 ,2.8919160E-38 ,4.5833780E-38 ,7.2641646E-38,
     + 1.1512925E-37, 1.8246757E-37, 2.8919161E-37, 4.5833782E-37,
     + 7.2641649E-37, 1.1512925E-36, 1.8246757E-36, 2.8919161E-36,
     + 4.5833782E-36, 7.2641649E-36, 1.1512925E-35, 1.8246757E-35,
     + 2.8919161E-35, 4.5833782E-35, 7.2641649E-35, 1.1512925E-34,
     + 1.8246757E-34, 2.8919161E-34, 4.5833782E-34, 7.2641649E-34,
     + 1.1512925E-33, 1.8246757E-33, 2.8919161E-33, 4.5833782E-33,
     + 7.2641649E-33, 1.1512925E-32, 1.8246757E-32, 2.8919161E-32,
     + 4.5833782E-32, 7.2641649E-32, 1.1512925E-31, 1.8246757E-31,
     + 2.8919161E-31, 4.5833782E-31, 7.2641649E-31, 1.1512925E-30,
     + 1.8246757E-30, 2.8919161E-30, 4.5833782E-30, 7.2641649E-30,
     + 1.1512925E-29, 1.8246757E-29, 2.8919161E-29, 4.5833782E-29,
     + 7.2641649E-29, 1.1512925E-28, 1.8246757E-28, 2.8919161E-28,
     + 4.5833782E-28, 7.2641649E-28, 1.1512925E-27, 1.8246757E-27,
     + 2.8919161E-27, 4.5833782E-27, 7.2641649E-27, 1.1512925E-26,
     + 1.8246757E-26, 2.8919161E-26, 4.5833782E-26, 7.2641649E-26,
     + 1.1512925E-25, 1.8246757E-25, 2.8919161E-25, 4.5833782E-25,
     + 7.2641649E-25, 1.1512925E-24, 1.8246757E-24, 2.8919161E-24,
     + 4.5833782E-24, 7.2641649E-24, 1.1512925E-23, 1.8246757E-23,
     + 2.8919161E-23, 4.5833782E-23, 7.2641649E-23, 1.1512925E-22,
     + 1.8246757E-22, 2.8919161E-22, 4.5833782E-22, 7.2641649E-22,
     + 1.1512925E-21, 1.8246757E-21, 2.8919161E-21, 4.5833782E-21,
     + 7.2641649E-21, 1.1512925E-20, 1.8246757E-20, 2.8919161E-20,
     + 4.5833782E-20, 7.2641649E-20, 1.1512925E-19, 1.8246757E-19,
     + 2.8919161E-19, 4.5833782E-19, 7.2641649E-19, 1.1512925E-18,
     + 1.8246757E-18, 2.8919161E-18, 4.5833782E-18, 7.2641649E-18,
     + 1.1512925E-17, 1.8246757E-17, 2.8919161E-17, 4.5833782E-17,
     + 7.2641649E-17, 1.1512925E-16, 1.8246757E-16, 2.8919161E-16,
     + 4.5833782E-16, 7.2641649E-16, 1.1512925E-15, 1.8246757E-15,
     + 2.8919161E-15, 4.5833782E-15, 7.2641649E-15, 1.1512925E-14,
     + 1.8246757E-14, 2.8919161E-14, 4.5833782E-14, 7.2641649E-14,
     + 1.1512925E-13, 1.8246757E-13, 2.8919161E-13, 4.5833782E-13,
     + 7.2641649E-13, 1.1512925E-12, 1.8246757E-12, 2.8919161E-12,
     + 4.5833782E-12, 7.2641649E-12, 1.1512925E-11, 1.8246757E-11,
     + 2.8919161E-11, 4.5833782E-11, 7.2641649E-11, 1.1512925E-10,
     + 1.8246757E-10, 2.8919161E-10, 4.5833782E-10, 7.2641649E-10,
     + 1.1512925E-09, 1.8246757E-09, 2.8919161E-09, 4.5833782E-09,
     + 7.2641648E-09, 1.1512925E-08, 1.8246757E-08, 2.8919160E-08,
     + 4.5833779E-08, 7.2641643E-08, 1.1512924E-07, 1.8246754E-07,
     + 2.8919152E-07, 4.5833759E-07, 7.2641591E-07, 1.1512911E-06,
```

```
130       + 1.8246721E-06, 2.8919071E-06, 4.5833553E-06, 7.2641076E-06,
131       + 1.1512781E-05, 1.8246396E-05, 2.8918252E-05, 4.5831503E-05,
132       + 7.2635914E-05, 1.1511488E-04, 1.8243139E-04, 2.8910090E-04,
133       + 4.5810955E-04, 7.2584422E-04, 1.1498526E-03, 1.8210664E-03,
134       + 2.8828366E-03, 4.5606276E-03, 7.2069676E-03, 1.1369739E-02,
135       + 1.7887297E-02, 2.8021393E-02, 4.3588251E-02, 6.7066119E-02,
136       + 1.0131146E-01, 1.4866030E-01, 2.0732317E-01, 2.6530137E-01,
137       + 2.8524462E-01, 2.0254407E-01,-5.7375154E-02,-3.7135384E-01,
138       +-3.1963734E-01, 4.6955122E-01,-2.7699099E-02,-2.0391968E-01,
139       + 1.8174329E-01,-1.0118230E-01, 4.4856879E-02,-1.8088601E-02,
140       + 7.1995987E-03,-2.8661435E-03, 1.1410288E-03,-4.5425151E-04,
141       + 1.8084077E-04,-7.1994008E-05, 2.8661331E-05,-1.1410281E-05,
142       + 4.5425148E-06,-1.8084077E-06, 7.1994007E-07,-2.8661331E-07,
143       + 1.1410281E-07,-4.5425148E-08, 1.8084077E-08,-7.1994007E-09,
144       + 2.8661331E-09,-1.1410281E-09, 4.5425148E-10,-1.8084077E-10,
145       + 7.1994007E-11,-2.8661331E-11, 1.1410281E-11,-4.5425148E-12,
146       + 1.8084077E-12,-7.1994007E-13, 2.8661331E-13,-1.1410281E-13,
147       + 4.5425148E-14,-1.8084077E-14, 7.1994007E-15,-2.8661331E-15,
148       + 1.1410281E-15,-4.5425148E-16, 1.8084077E-16,-7.1994007E-17,
149       + 2.8661331E-17,-1.1410281E-17, 4.5425148E-18,-1.8084077E-18,
150       + 7.1994007E-19,-2.8661331E-19, 1.1410281E-19,-4.5425148E-20,
151       + 1.8084077E-20,-7.1994007E-21, 2.8661331E-21,-1.1410281E-21,
152       + 4.5425148E-22,-1.8084077E-22, 7.1994007E-23,-2.8661331E-23,
153       + 1.1410281E-23,-4.5425148E-24, 1.8084077E-24,-7.1994007E-25,
154       + 2.8661331E-25,-1.1410281E-25, 4.5425148E-26,-1.8084077E-26,
155       + 7.1994007E-27,-2.8661331E-27, 1.1410281E-27,-4.5425148E-28,
156       + 1.8084077E-28,-7.1994007E-29, 2.8661331E-29,-1.1410281E-29,
157       + 4.5425148E-30,-1.8084077E-30, 7.1994007E-31,-2.8661331E-31,
158       + 1.1410281E-31,-4.5425148E-32, 1.8084077E-32,-7.1994007E-33,
159       + 2.8661331E-33,-1.1410281E-33, 4.5425148E-34,-1.8084077E-34,
160       + 7.1994007E-35,-2.8661331E-35, 1.1410281E-35,-4.5425148E-36,
161       + 1.8084077E-36/
```

`initw` is an initiation routine, called for each frequency, that computes the admittivity, impedivity and coefficient of anisotropy for each layer of interest.

Listing of `initw.f`

```
1  c-----------------------------------------------------------------------
2        subroutine initw(sigmah,sigmav,epsiln,freq)
3  c-----------------------------------------------------------------------initw
4  c sets up the parameters for the common blocks
5  c
6        include  'dipole.inc'
7        integer i
8        real*8  epsiln(maxlay), sigmah(maxlay), sigmav(maxlay)
9        real*8  omega, freq
10 c
11       omega = freq * 2.0d0 * pi
12       do i = 1, layers
13         alphah(i) = dcmplx(sigmah(i),omega*epsiln(i))
14         alphav(i) = dcmplx(sigmav(i),omega*epsiln(i))
15         kappa(i)  = alphah(i)/alphav(i)
16         beta(i)   = dcmplx(0.0d0,omega*mu(i))
17       end do
18       return
19       end
```

`initd` is called only when the layered earth model changes and computes the interface depths and distances between the source and receiver and the nearest interfaces.

Listing of `initd.f`

```
1  c-----------------------------------------------------------------------
```

```
      subroutine initd
c-----------------------------------------------------------------------initd
      include  'dipole.inc'
      integer i
      real*8  depth(maxlay)
c
c find the relative positions of the layers in the earth
c also locate the layers containing the source and receiver
c
      r = 1
      s = 1
      depth(1) = 0.0d0
      do i = 2, layers-1
        depth(i) = depth(i-1) + thckns(i)
        if(rz.le.depth(i).and.rz.gt.depth(i-1)) r = i
        if(sz.le.depth(i).and.sz.gt.depth(i-1)) s = i
      end do
      if(rz.gt.depth(layers-1)) r = layers
      if(sz.gt.depth(layers-1)) s = layers
      write(6,*) 'Source layer = ',s,';  Receiver layer = ',r
c
c find the distance from the source to the nearset interface
c above and below the source level ( required for the reflection
c coefficients
c
      if(s.ne.1)      sdepup = abs(sz - depth(s-1))
      if(s.ne.layers) sdepdn = abs(sz - depth(s))
c
      if(r.ne.s) then
        if(rz.ge.sz) then
          sdist = sdepdn
          rdist = abs(rz - depth(r-1))
         else if(rz.lt.sz) then
          sdist = sdepup
          rdist = abs(rz - depth(r))
        end if
       else if(r.eq.s) then
        sdist = abs(sz - rz)
        rdist = sdist
      end if
      return
      end
```

`orient` finds the geometrical orientation factors for an arbitrary dipole at an angle of θ with respect to the x-axis.

Listing of `orient.f`

```
c-----------------------------------------------------------------------
      subroutine orient(wc,theta)
c-----------------------------------------------------------------------orient
c This subroutine generates the geometrical orientation factors for the
c arbitrary dipole problem with an angle theta.
c FACTOR is 1/(4 pi) and theta is measured from the axis of the dipole.
c Boerner & West, 1989, Geophysical Journal, 97, 529-547.
c
      real*8 factor, costt, sintt, theta, wc(2,5)
      parameter (factor = 0.07957747154594768)
c
      costt = cosd( 2.0d0 * theta )
      sintt = sind( 2.0d0 * theta )
c
      wc(1,1) = -factor * ( costt + 1.0d0 ) / 2.0d0
      wc(1,2) = -factor * sintt / 2.0d0
      wc(1,3) =  0.0d0
      wc(1,4) =  0.0d0
```

```
      wc(1,5) =  factor
c
      wc(2,1) =  factor * costt
      wc(2,2) =  factor * sintt
      wc(2,3) = -factor * cosd( theta )
      wc(2,4) = -factor * sind( theta )
      wc(2,5) =  0.0d0
c
      return
      end
```

jsrch finds the total field kernels due to electric dipole sources as expressed in the Hankel domain. The components calculated are essentially those given in Table 1a.

Listing of jsrch.f

```
c-----------------------------------------------------------------------
      subroutine jsrch(u,v,lambda2,t,p,hfield)
c-----------------------------------------------------------------------
c total fields due to electric sources in Hankel Domain.
c
      include  'dipole.inc'
c
      complex*16 u(maxlay), v(maxlay), p(4), t(4), hfield(6,5)
      complex*16 ej(5), hj(4), rat
      real*8     lambda2
c
      rat = kappa(s) * v(r) / ( kappa(r) * v(s) )
      ej(1) =            v(r) * p(3) / ( alphav(s) * kappa(r) )
      ej(2) =         beta(r) * t(1) / u(s)
      ej(3) =             rat * p(2) / alphav(s)
      ej(4) = lambda2 * v(r) * p(1) / ( alphav(s) * kappa(r) )
      ej(5) =                  p(4) / alphav(s)
c
      hfield(1,1) = (ej(1)-ej(2))*js(1)
      hfield(1,2) = (ej(1)-ej(2))*js(2)
      hfield(1,3) =  ej(3)*js(3)
      hfield(1,4) = cmplx(0.0,0.0)
      hfield(1,5) = -ej(2)*js(1)
c
      hfield(2,1) =-(ej(1)-ej(2))*js(2)
      hfield(2,2) = (ej(1)-ej(2))*js(1)
      hfield(2,3) = cmplx(0.0,0.0)
      hfield(2,4) =  ej(3)*js(3)
      hfield(2,5) = -ej(1)*js(2)
c
      hfield(3,1) = cmplx(0.0,0.0)
      hfield(3,2) = cmplx(0.0,0.0)
      hfield(3,3) =  ej(5)*js(1)
      hfield(3,4) =  ej(5)*js(2)
      hfield(3,5) = -ej(4)*js(3)
c
c Magnetic field components
c
      rat = alphav(r)/alphav(s)
      hj(1) =           u(r) * t(2) / u(s)
      hj(2) =            rat * p(4)
      hj(3) = rat * kappa(s) * p(1) / v(s)
      hj(4) =                  t(1) / u(s)
c
      hfield(4,1) = (hj(2)-hj(1))*js(2)
      hfield(4,2) =-(hj(2)-hj(1))*js(1)
      hfield(4,3) = cmplx(0.0,0.0)
      hfield(4,4) =  hj(3)*js(3)
```

```
      hfield(4,5) =  hj(2)*js(2)
c
      hfield(5,1) = (hj(2)-hj(1))*js(1)
      hfield(5,2) = (hj(2)-hj(1))*js(2)
      hfield(5,3) = -hj(3)*js(3)
      hfield(5,4) = cmplx(0.0,0.0)
      hfield(5,5) = -hj(1)*js(1)
c
      hfield(6,1) = cmplx(0.0,0.0)
      hfield(6,2) = cmplx(0.0,0.0)
      hfield(6,3) = -hj(4)*js(2)
      hfield(6,4) =  hj(4)*js(1)
      hfield(6,5) = cmplx(0.0,0.0)
c
      return
      end
```

msrch finds the total field kernels due to magnetic dipole sources as expressed in the Hankel domain. The components calculated are essentially those given in Table 1b.

Listing of msrch.f

```
c-----------------------------------------------------------------------
      subroutine msrch(u,v,lambda2,t,p,hfield)
c-----------------------------------------------------------------------
c total fields due to magnetic sources in Hankel Domain.
c
      include  'dipole.inc'
c
      complex*16 u(maxlay), v(maxlay), p(4), t(4), hfield(6,5)
      complex*16 rat, ej(4), hj(5)
      real*8     lambda2
c
      rat = kappa(s) / kappa(r)
      ej(1) =-rat * v(r) * p(2) * beta(s) / v(s)
      ej(2) =              t(4) * beta(r)
      ej(3) =              t(1) * beta(r) / u(s)
      ej(4) =   kappa(s) * p(1) * beta(s) / v(s)
c
      hfield(1,1) =-(ej(1)+ej(2))*ms(2)
      hfield(1,2) = (ej(1)+ej(2))*ms(1)
      hfield(1,3) = cmplx(0.0,0.0)
      hfield(1,4) =  ej(3)*ms(3)
      hfield(1,5) = -ej(2)*ms(2)
c
      hfield(2,1) =-(ej(1)+ej(2))*ms(1)
      hfield(2,2) =-(ej(1)+ej(2))*ms(2)
      hfield(2,3) = -ej(3)*ms(3)
      hfield(2,4) = cmplx(0.0,0.0)
      hfield(2,5) = -ej(1)*ms(1)
c
      hfield(3,1) = cmplx(0.0,0.0)
      hfield(3,2) = cmplx(0.0,0.0)
      hfield(3,3) = -ej(4)*ms(2)
      hfield(3,4) =  ej(4)*ms(1)
      hfield(3,5) = cmplx(0.0,0.0)
c
c Magnetic field components
c
      hj(1) =                 u(r) * t(3)
      hj(2) =-alphav(r) * kappa(s) * p(1) * beta(s) / v(s)
      hj(3) =                 u(r) * t(2) / u(s)
      hj(4) =                        t(4)
```

```
42        hj(5) =              lambda2 * t(1) / u(s)
43  c
44        hfield(4,1) = (hj(2)+hj(1))*ms(1)
45        hfield(4,2) = (hj(2)+hj(1))*ms(2)
46        hfield(4,3) =  hj(3)*ms(3)
47        hfield(4,4) = cmplx(0.0,0.0)
48        hfield(4,5) =  hj(2)*ms(1)
49  c
50        hfield(5,1) =-(hj(2)+hj(1))*ms(2)
51        hfield(5,2) = (hj(2)+hj(1))*ms(1)
52        hfield(5,3) = cmplx(0.0,0.0)
53        hfield(5,4) =  hj(3)*ms(3)
54        hfield(5,5) = -hj(1)*ms(2)
55
56        hfield(6,1) = cmplx(0.0,0.0)
57        hfield(6,2) = cmplx(0.0,0.0)
58        hfield(6,3) =  hj(4)*ms(1)
59        hfield(6,4) =  hj(4)*ms(2)
60        hfield(6,5) = -hj(5)*ms(3)
61  c
62        return
63        end
```

look is a simple I/O routine that echos the input information out to the screen to check the veracity of the input data file.

Listing of look.f

```
 1  c-----------------------------------------------------------------------
 2        subroutine look(maxlay,layers,sigmah,sigmav,epsiln,mu,
 3       >                thckns,freq1,freqmx,den,maxrad,numrad,radii,
 4       >                ntheta,theta,tinc,js,ms,sz,rz)
 5  c-----------------------------------------------------------------------
 6  c look at the input parameters
 7  c
 8        character*2 chars(6)
 9        integer i, ij, layers, numrad, maxlay, maxrad, ntheta
10        real*8  freq1,freqmx,sigmah(maxlay),sigmav(maxlay), tinc
11        real*8  epsiln(maxlay), js(3), ms(3), mu(maxlay), jms(3)
12        real*8  theta, radii(maxrad), den, thckns(maxlay), sz, rz
13        data    chars/'Jx','Jy','Jz','Mx','My','Mz'/
14  c
15        ij = 1
16        jms(1) = js(1)
17        jms(2) = js(2)
18        jms(3) = js(3)
19        if(ms(1).ne.0.or.ms(2).ne.0.or.ms(3).ne.0) then
20          ij = 4
21          jms(1) = ms(1)
22          jms(2) = ms(2)
23          jms(3) = ms(3)
24        end if
25  c
26        write(6,1) freq1, freqmx, den, chars(ij), chars(ij+1),
27       >           chars(ij+2), jms(1), jms(2), jms(3), sz, rz,
28       >           numrad, radii(1), ntheta, theta, tinc
29  1     format(' frequency range    ; ',1pe9.3,' to ',1pe9.3,' Hz',/,
30       >       '                    ; ',0pf5.2,' points per decade',/,
31       >       ' source component   ; ',2x,3(5x,a2,5x),/,
32       >       '                    ; ',2x,3(1pe12.4),/,
33       >       '  Tx/Rx height (m)  ; ',1pe11.3,' / ',1pe11.3,/,
34       >       ' # and initial r    ; ',i6,     ' / ',1pe11.3,/,
```

```
35       >       ' # and initial theta; ',i6,      ' / ',0pf7.2,/,
36       >       '  theta increment    ; ',0pf7.2,/)
37  c
38        write(6,2)
39  2     format(/,2x,'#', 3x,'sigma_h', 4x,'sigma_v',
40       >       3x,'epsilon',7x,'mu',9x,'tau'     ,7x,'thckns'/,
41       >       6x,' (S/m) ', 4x,' (S/m) ',
42       >       4x,'(F/m)',7x,'(H/m)',7x,'  (m) '/)
43        i=0
44        do i=1,layers
45          if(i.ne.layers.and.i.ne.1) then
46            if(sigmav(i).eq.sigmah(i)) then
47              write(6,5) i,sigmah(i),sigmav(i),epsiln(i),mu(i),thckns(i)
48             else
49              write(6,7) i,sigmah(i),sigmav(i),epsiln(i),mu(i),thckns(i)
50            end if
51           else
52            if(sigmav(i).eq.sigmah(i)) then
53              write(6,3) i,sigmah(i),sigmav(i),epsiln(i),mu(i)
54             else
55              write(6,8) i,sigmah(i),sigmav(i),epsiln(i),mu(i)
56            end if
57          end if
58          if(i.ne.layers) write(6,4)
59        end do
60        write(6,6)
61  3     format(1x,i2,5(1x,1pe10.3),1x,'  Infinite ')
62  4     format(1x,72('-'))
63  5     format(1x ,i2,6(1x,1pe10.3))
64  6     format(//)
65  7     format('K',i2,5(1x,1pe10.3))
66  8     format('K',i2,4(1x,1pe10.3))
67        return
68        end
```

reader reads the input data file.

Listing of reader.f

```
1   c-----------------------------------------------------------------------
2         subroutine reader(maxlay,layers,sigmah,sigmav,epsiln,mu,
3        >                  thckns,freq1,ndec,den,maxrad,numrad,radii,
4        >                  ntheta,theta,tinc,js,ms,sz,rz,tol,fht_type)
5   c-----------------------------------------------------------------------
6   c this file reads the required data from an existing data file
7   c
8         integer i, ndec, layers, numrad, maxlay, maxrad, ntheta
9         integer st1, fht_type
10        real*8 thckns(maxlay), den, epsiln(maxlay)
11        real*8 mu(maxlay), freqn, freq1, sx, sy, idl
12        real*8 sigmav(maxlay), sigmah(maxlay)
13        real*8 radii(maxrad), theta, sz, rz, eps, js(3), ms(3)
14        real*8 saz, sin, ds, eps_rel
15        real*8 mu_rel, mufs, tol, rden, rmax, tmax, tinc
16        character*80 name, line
17  c
18        parameter( eps = 8.8542090d-12,
19       >           mufs  = 1.25663706143591729d-06 )
20  c
21        name='dipole.dat'
22        open(unit=1,file=name,status='old')
23
24        call get_line(line,*10)
25        read(line,*) tol, fht_type
26        call get_line(line,*10)
```

```
27        read(line,*) st1
28        call get_line(line,*10)
29        read(line,*) sx, sy, sz
30        call get_line(line,*10)
31        read(line,*) idl, saz, sin, ds
32        if ( st1.eq.0 ) then
33          js(1) = idl * cosd( saz ) * cosd( sin )
34          js(2) = idl * sind( saz ) * cosd( sin )
35          js(3) = idl * sind( sin )
36         else
37          ms(1) = ds * cosd( saz ) * cosd( sin )
38          ms(2) = ds * sind( saz ) * cosd( sin )
39          ms(3) = ds * sind( sin )
40        end if
41  c
42        call get_line(line,*10)
43        read(line,*) freq1, freqn, den
44        ndec = nint(dlog10(freqn/freq1))
45        call get_line(line,*10)
46        read(line,*) radii(1), rmax, rden, rz, theta, tmax, tinc
47        numrad = max(int(dlog10(rmax/radii(1))*rden),1)
48        ntheta = max(int((tmax - theta)/tinc),1)
49  c
50        call get_line(line,*10)
51        read(line,*) layers
52        layers = layers + 2
53  c
54        call get_line(line,*10)
55        read(line,*) sigmah(1), sigmav(1), eps_rel, mu_rel
56        epsiln(1) = eps  * eps_rel
57        mu(1)     = mufs *  mu_rel
58  c
59        do i = 2, layers-1
60          call get_line(line,*10)
61          read(line,*) thckns(i), sigmah(i), sigmav(i),
62       >               eps_rel, mu_rel
63          epsiln(i) = eps  * eps_rel
64          mu(i)     = mufs * mu_rel
65        end do
66  c
67        call get_line(line,*10)
68        read(line,*) sigmah(layers), sigmav(layers),
69       >              eps_rel, mu_rel
70        epsiln(layers) = eps  * eps_rel
71        mu(layers)     = mufs *  mu_rel
72  c
73  10    return
74        end
```

`matmult` multiplies two 2×2 matrices together.

Listing of `matmult.f`

```
1   c-----------------------------------------------------------------------
2         subroutine matmult(a,b,c)
3   c-----------------------------------------------------------------------
4   c multiply a and b (2x2 matrices) together and store the
5   c result in c.
6   c
7         complex*16 a(2,2), b(2,2), c(2,2), temp(2,2)
8   c
9         temp(1,1) = a(1,1) * b(1,1) + a(1,2) * b(2,1)
10        temp(1,2) = a(1,1) * b(1,2) + a(1,2) * b(2,2)
11        temp(2,1) = a(2,1) * b(1,1) + a(2,2) * b(2,1)
12        temp(2,2) = a(2,1) * b(1,2) + a(2,2) * b(2,2)
```

```
13  c
14        c(1,1) = temp(1,1)
15        c(1,2) = temp(1,2)
16        c(2,1) = temp(2,1)
17        c(2,2) = temp(2,2)
18  c
19        return
20        end
```

itoa converts an integer to an ascii character.

Listing of itoa.f

```
 1  c-----------------------------------------------------------------------
 2        subroutine itoa(char,num,i)
 3  c-----------------------------------------------------------------------
 4        character char*(*)
 5        integer num, i
 6
 7        if(num.le.9) then
 8          write(char(1:1),'(i1)') num
 9          i = 1
10        end if
11        if(num.gt.9.and.num.lt.100) then
12          write(char(1:2),'(i2)') num
13          i = 2
14        end if
15        return
16        end
```

get_line reads a line from the input control file and ignores it if the line begins with a comment character (#).

Listing of getline.f

```
 1  c-----------------------------------------------------------------------
 2        subroutine get_line(line,*)
 3  c-----------------------------------------------------------------------
 4        character*(*) line
 5
 6  100   read(1,'(a)',end=10) line
 7        if(line(1:1).eq.'#') goto 100
 8        if(line.eq.' ') goto 100
 9        return
10  10    return 1
11        end
```

The following listing shows an example of the input data file format.

Listing of dipole.dat.example

```
 1  # - all lines begining with # are ignored
 2  # FHT definitions, tolerance = relative error
 3  # (type: 1 = Convolution, 2 = Direct integration )
 4  # tolerance   type
 5    1.0e-08      1
 6  #
 7  # 0 = Electric Dipole, 1 = Magnetic Dipole
 8  #
 9    1
10  #
11  # source location (meters)
12  # src_x    src_y        src_z
```

```
#
  0.0       0.0           -0.01
#
# Idl (Amps)   source azimuth   source inclination     I dS (A/m)
#
  1.0          0.0              90.0                   1.0
#
# freq_low freq_high     nfreq/decade
#
  1.e-02    1.0e+07            8
#
# Receiver profile definitions (rz = depth of receiver, +ve down) :
# also calculate on angles sweeping from ang1 to ang2 in increments of anginc
# rmin    rmax    r/dec. rz    ang1  ang2  anginc
#
500.0     500.0  10     0.0  30.0     0.0  5
#
# Model definition:
# number of layers
  1
# upper half space
#          hcond.     vcond     eps_rel    mu_rel
           1.0e-14    1.0e-14     0.0        1.0
# layers
# thck     hcond.     vcond     eps_rel    mu_rel
  250.0     0.235      0.235      1.0        1.0
# lower half space
#          hcond.     vcond     eps_rel    mu_rel
            0.01       0.01       1.0        1.0
```

4 Testing the Software

One of the most reliable checks on the software being compiled properly and producing valid result is to demonstrate reciprocity. This involves computing the fields from a given source and then interchanging the source and receiver and recomputing the fields. Reciprocity says that in a linear medium (i.e., a medium exhibiting no dispersive properties), the source and receiver are exactly reciprocal. A `tcl` script file has been provided along with the code to check automatically that reciprocity is valid for each source-receiver pair.

`tcl`, tool command language, is a multi-purpose scripting language available for many computer systems (e.g. UNIX, Windows95, etc. It is freely available in both source and binary distributions from `http://sunscript.sun.com/` . The path of the `emdipole` executable must be specified in the script files and then run recip.tcl to output the real and imaginary parts of the EM fields as well as the error in the reciprocity calculation. Reciprocity is quite a stringent test since it exercises all parts of the propagation matrix calculation, the reflection coefficients and the source matrices. Furthermore, depending upon the parameters of the layered earth model for which it is run, reciprocity also tests the implementation of the transverse anisotropy derivation.

The following is an example of the `recip.tcl` script file.

Listing of `recip.tcl`

```
#!/apps/local/bin/wish
#
# Test Reciprocity
```

```
4  # - note that to check reciprocity between electric and magnetic
5  # fields you need to normalize by i omega mu_R. This script only
6  # normalizes by i omega mu_freespace.
7  #
8  set PROGRAM ./emdipole
9  #
10 # set test parameters -
11 # FRQ = test frequency
12 # RAD = source/receiver separation
13 # ANG = angle between the x axis and the receiver
14 # SDP = source depth
15 # RDP = receiver depth
16 #
17 set FRQ   1.00e-05
18 set RAD   100.00
19 set ANG    60.00
20 set SDP    35.10
21 set RDP     0.01
22
23 proc main {} {
24   global FRQ FRH RAD ANG SDP RDP COPY
25
26   puts "\nPercentage difference in real and imaginary parts\
27         as determined from reciprocity calculations.\n"
28   puts "          REAL                 IMAGINARY          Pair  %Error Re %Error Im"
29   set FRH [expr $FRQ * 5.0]
30 #
31 # Compute only non-redundant source-receiver pairs.
32 #
33   set com { exex exey exez exhx exhy exhz
34                  eyey eyez eyhx eyhy eyhz
35                       ezez ezhx ezhy ezhz
36                            hxhx hxhy hxhz
37                                 hyhy hyhz
38                                      hzhz }
39
40   foreach pair $com {
41     set src [string range $pair 0 1]
42     set fld [string range $pair 2 3]
43
44     set dir [string index $pair 1]
45     set typ [string index $pair 0]
46
47     set SIN 0.0
48     set SAZ 0.0
49     set DIP 0
50     if { $dir == "y" } { set SAZ 90.0 }
51     if { $dir == "z" } { set SIN 90.0 }
52     if { $typ == "h" } { set DIP 1 }
53     run  $SDP $RDP $DIP $SAZ $SIN
54     read $fld a b
55 #
56 # Reverse the source and receiver parameters
57 #
58     set dir [string index $pair 3]
59     set typ [string index $pair 2]
60
61     set SIN 0.0
62     set SAZ 0.0
63     set DIP 0
64     if { $dir == "y" } { set SAZ 90.0 }
65     if { $dir == "z" } { set SIN 90.0 }
66     if { $typ == "h" } { set DIP 1 }
67
68     run  $RDP $SDP $DIP $SAZ $SIN
69     read $src c d
70 #
71 # print out comparison
72 #
73     puts [format "(%10.3e %10.3e) (%10.3e %10.3e) - $pair: %.3e %.3e " \
```

```
74                $a $c $b $d [error $a $c] [error $b $d]]
75    }
76  }
77  proc error {a b} {
78    if { abs($b) != 0.0 } {
79      return [expr 100*(abs($a-$b))/abs($b)]
80    } else {
81      return 0.0
82    }
83  }
84
85  proc run { SDP RDP DIP SAZ SIN } {
86    global ANG RAD FRQ FRH PROGRAM
87  #
88  # write a data file with the appropriate parameters
89  #
90    set f [ open dipole.dat w ]
91      puts $f "1.0e-10 1"
92      puts $f "$DIP"
93      puts $f "0.0 0.0 $SDP"
94      puts $f "1.0 $SAZ $SIN 1.0"
95      puts $f "$FRQ $FRH 2 0 0.0"
96      puts $f "$RAD $RAD 10 $RDP $ANG $ANG 5"
97  #
98  # specify the layered earth model (layers + two terminating halfspaces)
99  #
100     puts $f "1"
101 #
102 # -- air
103 # -- layer parameters (repeat as necessary) (sh, sv, epsilon, mu)
104 # -- lower halfspace
105 #
106     puts $f "    1.0e-14 1.0e-14  1.0 1.0"
107     puts $f "30  1.0e-01 1.0e-05 80.0 1.0"
108     puts $f "    1.0e-05 1.0e-01  1.0 1.0"
109   close $f
110   exec $PROGRAM >& /dev/null
111 }
112 proc read { fld real imag } {
113   upvar $real r
114   upvar $imag i
115   global FRQ
116
117   set con 7.89568352087148689e-06
118   set out ""
119   set f [ open $fld.fld r ]
120   while { [gets $f line ] >= 0 && $out == "" } {
121     regsub D [lindex $line 0] e input
122     if { $FRQ == $input } {
123      set out $line
124     }
125   }
126   close $f
127 #
128 # change the D in the double precision fortran output to an e
129 #
130   regsub D [lindex $out 1] e re
131   regsub D [lindex $out 2] e im
132 #
133 # If it is a magnetic field, we need to convert
134 #
135   if [string match *h* $fld] {
136     set r [expr -$con * $im * $FRQ]
137     set i [expr  $con * $re * $FRQ]
138    } else {
139     set r $re
140     set i $im
```

```
141     }
142   }
143   main
144   exit
```

`recip.tcl` produces screen output showing the calculated fields and a measure of the percent difference created by exchanging the source and receiver. An example output follows based on model contained in the `recip.tcl` script shown above.

Listing of `recip.out`

```
1
2  Percentage difference in real and imaginary parts as determined
3  from reciprocity calculations.
4
5            REAL                    IMAGINARY            Pair  %Error Re %Error Im
6  (-5.880e-09 -5.880e-09) (-9.159e-14 -9.159e-14) - exex: 0.000e+00 0.000e+00
7  ( 8.663e-12  8.663e-12) (-1.289e-14 -1.289e-14) - exey: 0.000e+00 0.000e+00
8  (-3.913e-11 -3.913e-11) (-6.043e-17 -6.043e-17) - exez: 1.533e-11 1.324e-11
9  ( 1.527e-24  1.527e-24) ( 2.837e-16  2.837e-16) - exhx: 4.991e-12 9.036e-13
10 (-2.629e-24 -2.629e-24) ( 7.121e-17  7.121e-17) - exhy: 2.487e-12 4.501e-13
11 (-4.311e-24 -4.311e-24) (-4.571e-16 -4.571e-16) - exhz: 1.022e-13 1.079e-13
12 (-5.870e-09 -5.870e-09) (-1.065e-13 -1.065e-13) - eyey: 0.000e+00 0.000e+00
13 (-6.778e-11 -6.778e-11) (-1.047e-16 -1.047e-16) - eyez: 1.623e-11 9.550e-12
14 ( 4.393e-24  4.393e-24) ( 2.564e-16  2.564e-16) - eyhx: 6.188e-13 1.057e-12
15 (-1.527e-24 -1.527e-24) (-2.837e-16 -2.837e-16) - eyhy: 4.991e-12 9.036e-13
16 ( 2.489e-24  2.489e-24) ( 2.639e-16  2.639e-16) - eyhz: 8.412e-13 1.849e-12
17 (-2.589e-17 -2.589e-17) (-2.368e-23 -2.368e-23) - ezez: 1.932e-11 1.689e-11
18 (-1.057e-30 -1.057e-30) (-4.019e-25 -4.019e-25) - ezhx: 3.051e-11 1.479e-11
19 ( 6.102e-31  6.102e-31) ( 2.321e-25  2.321e-25) - ezhy: 3.132e-11 1.375e-11
20 (-0.000e+00  0.000e+00) ( 0.000e+00  0.000e+00) - ezhz: 0.000e+00 0.000e+00
21 (-3.034e-26 -3.034e-26) (-1.754e-18 -1.754e-18) - hxhx: 0.000e+00 0.000e+00
22 ( 3.358e-26  3.358e-26) ( 6.105e-18  6.105e-18) - hxhy: 0.000e+00 0.000e+00
23 (-3.125e-26 -3.125e-26) (-2.474e-18 -2.474e-18) - hxhz: 0.000e+00 0.000e+00
24 ( 8.432e-27  8.432e-27) ( 5.296e-18  5.296e-18) - hyhy: 0.000e+00 0.000e+00
25 (-5.413e-26 -5.413e-26) (-4.285e-18 -4.285e-18) - hyhz: 0.000e+00 0.000e+00
26 (-2.195e-26 -2.195e-26) ( 3.543e-18  3.543e-18) - hzhz: 0.000e+00 0.000e+00
```

Reciprocity is a necessary, but not sufficient test of the algorithm. Another useful test is to examine the boundary conditions by placing a receiver just above and just below every interface in the model to see if the continuity of tangential **E** and **H**, and normal **J** and **B** is maintained.

The reciprocity and boundary condition tests described above are important in verifying the code is working properly, but they do not guarantee the source fields are correct. To test the calculation more fully, it is important to compare the results with analytical expressions, or results obtained by other authors. Two particularly useful test suites are the analytical expressions for a wholespace (e.g. Ward and Hohmann, 1991), and for a uniform halfspace as provided by Bannister (1966). A `tcl` script file for computing two of the examples shown in Ward and Hohmann (1991) is also included in this distribution. The script file is called `ward.tcl` and it contains the reference to the original paper and the figures that the examples should replicate. The output file of `ward.tcl` is called `ward.out`.

Listing of `ward.tcl`

```
#!/apps/local/bin/wish
#
# Test to match examples from Ward and Hohmann
# "Electromagnetic Theory for Geophysical Applications" Chapter 4, pages 131-311
# Electromagnetic Methods in Applied Geophysics - Theory
# Edited by Misac N. Nabighian, Society of Exploration Geophysicists
# 1991
#
set PROGRAM ./emdipole
#
# OUTPUT FILES = ward.out (containing real and imaginary parts)
#
# - You shouldn't change these constants for the comparison tests
#
set FRQ     1.0e-2
set FRH     3.5e+5
set RAD   100.00
set SAZ     0.00
set SDP     0.01
set RDP    -0.01
set SIGMA   1.0e-02
set METHOD  1
set SIN     0.0
set DIP     1
set fld     hx
#
# Comment out the tests you don't want
#
# fg2 -Hx equitorial wholespace: page 176 - fig 2-2
# fg3 -Hx coaxial   wholespace  :page 177 - fig 2-3
#set type fg3
set type fg2

if { $type == "fg2" } { set ANG 90.0 }
if { $type == "fg3" } { set ANG  0.0 }

proc main {} {
  global FRQ FRH RAD ANG SDP RDP COPY DIP SAZ SIN outfile fld
  global SIGMA
#
  set outfile [ open ward.out w ]

  run $SIGMA
  read $fld $SIGMA
}

proc run { SIGMA } {
  global ANG RAD FRQ FRH SDP RDP DIP SAZ SIN METHOD PROGRAM
#
# write a data file with the appropriate parameters
#
  set f [ open dipole.dat w ]
    puts $f "1.0e-10 $METHOD"
    puts $f "$DIP"
    puts $f "0.0 0.0 $SDP"
    puts $f "1.0 $SAZ $SIN 1"
    puts $f "$FRQ $FRH 10 0 0.0"
    puts $f "$RAD $RAD 10 $RDP $ANG $ANG 5"
#
# specify the layered earth model (two terminating halfspaces)
#
    puts $f "0"
#
# -- air
# -- lower halfspace
#
    puts $f "    $SIGMA  $SIGMA  1.0 1.0 0.0"
    puts $f "    $SIGMA  $SIGMA  1.0 1.0 0.0"
  close $f
  exec $PROGRAM >& /dev/null
}
```

```
proc read { fld SIGMA } {
  global DIP RAD outfile con

  set f [ open $fld.fld r ]
  while { [gets $f line ] >= 0 } {
    if { [string match @* $line] || \
         [string match #  [lindex $line 0]] || \
         [string match >  [lindex $line 0]] } {
      puts $outfile $line
    } else {
      regsub D [lindex $line 0] e freq
      regsub D [lindex $line 1] e re
      regsub D [lindex $line 2] e im
      set re [expr abs($re)]
      set im [expr abs($im)]
      puts $outfile [format "%15.8e %15.8e %15.8e" $freq $re $im]
    }
  }
  close $f
}
main
exit
```

Another suite of results involving layered models appears in the paper by Spies and Frischknecht (1991). The `tcl` script `spies.tcl` creates output files `spies.amp` and `spies.pha` using the program `emdipole` to compute the model examples presented by Spies and Frischknecht (1991) for comparison purposes.

Listing of `spies.tcl`

```
#!/apps/local/bin/wish
#
# Test to match examples from Spies and Fischknecht:
# "Electromagnetic Sounding" Chapter 5, pages 285-425
# Electromagnetic Methods in Applied Geophysics - Applications Part A
# Edited by Misac N. Nabighian, Society of Exploration Geophysicists
# 1991
#
set PROGRAM ./emdipole
#
# OUTPUT FILES = spies.amp spies.pha (containing amplitude and phase)
#
# - You shouldn't change these constants for the comparison tests
#
set FRQ     0.01
set FRH   100.00
set RAD  1000.00
set SAZ     0.00
set SDP     0.01
set RDP    -0.01
set SIGMA   2.35e-01
set METHOD 1
#
# Uncomment the test you want
#
# hcp -horizontal coplanar    :page 302 - fig 2-1 (a) and (b)
# vcp -vertical coplanar      :page 302 - fig 2-2 (a) and (b)
# vca -vertical coaxial       :page 302 - fig 2-3 (a) and (b)
# ehz -electric equitorial Hz :page 304 - fig 2-8 (a) and (b)
# ehy -electric equitorial Hy :page 304 - fig 2-9 (a) and (b)
# eex -electric equitorial Ex :page 305 - fig 2-11(a) and (b)
# ihy -electric inline Hy     :page 305 - fig 2-10(a) and (b)
# iex -electric inline Ex     :page 305 - fig 2-12(a) and (b)
```

```
#
# ppl -perpendicular loops    :page 303 - fig 2-4 (a) and (b)
#     -to do ppl calculation, you need to get the direct integration routine
#      by Alan Chave - see documentation
#
#set type ppl
#
set type hcp
#set type vcp
#set type vca
#set type ehz
#set type ehy
#set type ihy
#set type eex
#set type iex

if { $type == "hcp" } {
  set ANG     0.0
  set SIN    90.0
  set DIP     1
  set fld     hz
  set con [expr 3.14159 * 4.0 * $RAD * $RAD * $RAD]
}
if { $type == "vcp" } {
  set ANG    90.0
  set SIN     0.0
  set DIP     1
  set fld     hx
  set con [expr 3.14159 * 4.0 * $RAD * $RAD * $RAD]
}
if { $type == "vca" } {
  set ANG     0.0
  set SIN     0.0
  set DIP     1
  set fld     hx
  set con [expr 3.14159 * 4.0 * $RAD * $RAD * $RAD]
}
if { $type == "ppl" } {
  set ANG     0.0
  set SIN    90.0
  set DIP     1
  set fld     hx
  set con [expr 3.14159 * 4.0 * $RAD * $RAD * $RAD]
#
# FHT seems to fail in this particular case (probably because
# of poor convergence on the imaginary part) - use direct integration
#
  set METHOD 2
}
if { $type == "ehz" } {
  set ANG    90.0
  set SIN     0.0
  set DIP     0
  set fld     hz
  set con [expr 3.14159 * 4.0 * $RAD * $RAD]
}
if { $type == "ehy" } {
  set ANG    90.0
  set SIN     0.0
  set DIP     0
  set fld     hy
  set con [expr 3.14159 * 4.0 * $RAD * $RAD]
}
if { $type == "ihy" } {
  set ANG     0.0
  set SIN     0.0
  set DIP     0
  set fld     hy
  set con [expr 3.14159 * 4.0 * $RAD * $RAD]
}
if { $type == "eex" } {
```

```
105     set ANG    90.0
106     set SIN     0.0
107     set DIP     0
108     set fld     ex
109     set con [expr 3.14159 * 2.0 * $RAD * $RAD * $RAD * $SIGMA]
110   }
111   if { $type == "iex" } {
112     set ANG     0.0
113     set SIN     0.0
114     set DIP     0
115     set fld     ex
116     set con [expr 3.14159 * $RAD * $RAD * $RAD * $SIGMA]
117   }
118
119   proc main {} {
120     global FRQ FRH RAD ANG SDP RDP COPY DIP SAZ SIN amp_file pha_file fld
121     global SIGMA
122   #
123     set com { 0 0.3 1.0 3.0 10.0 30.0 100.0 }
124
125     set amp_file [ open spies.amp w ]
126     set pha_file [ open spies.pha w ]
127     foreach fact $com {
128       run $SIGMA $fact
129       read $fld $SIGMA
130     }
131   }
132
133   proc run { SIGMA fact } {
134     global ANG RAD FRQ FRH SDP RDP DIP SAZ SIN METHOD PROGRAM
135   #
136   # write a data file with the appropriate parameters
137   #
138     set SIGMA2 [expr $SIGMA*$fact]
139     set f [ open dipole.dat w ]
140       puts $f "1.0e-10 $METHOD"
141       puts $f "$DIP"
142       puts $f "0.0 0.0 $SDP"
143       puts $f "1.0 $SAZ $SIN 1.0"
144       puts $f "$FRQ $FRH 5"
145       puts $f "$RAD $RAD 10 $RDP $ANG $ANG 5"
146   #
147   # specify the layered earth model (layers + two terminating halfspaces)
148   #
149       puts $f "1"
150   #
151   # -- air
152   # -- first layer (repeat as necessary)
153   # -- lower halfspace
154   #
155       puts $f "    1.0e-14 1.0e-14 1.0 1.0 0.0"
156       puts $f "250 $SIGMA  $SIGMA  1.0 1.0 0.0"
157       puts $f "    $SIGMA2 $SIGMA2 1.0 1.0 0.0"
158     close $f
159     exec $PROGRAM >& /dev/null
160   }
161   proc read { fld SIGMA } {
162     global DIP RAD amp_file pha_file con
163
164
165     set f [ open $fld.fld r ]
166     while { [gets $f line ] >= 0 } {
167       if { [string match @* $line] || \
168            [string match #  [lindex $line 0]] || \
169            [string match >  [lindex $line 0]] } {
170         puts $amp_file $line
171         puts $pha_file $line
172       } else {
```

```
173         regsub D [lindex $line 0] e freq
174         regsub D [lindex $line 1] e re
175         regsub D [lindex $line 2] e im
176         puts $amp_file [format "%10.3e %10.3e"\
177               $freq [expr sqrt($re*$re+$im*$im)*$con]]
178         puts $pha_file [format "%10.3e %10.3e"\
179               $freq [expr atan2($im,$re)*180/3.14159]]
180       }
181    }
182    close $f
183 }
184 main
185 exit
```

5 Numerical Considerations

Although the exact expressions for the EM fields in a layered earth from a dipole source can always be written explicitly, there are some numerical limitations in evaluating these expressions. Some care should be used in computing fields with this program. In particular, numerical problems arise when the source or receiver are situated exactly on the same horizontal plane. (with the air/earth boundary presenting the worst case). It is generally prudent to separate the source and receiver by a millimeter or more to help ensure stability of the numerical calculation. This vertical offset is only important at very high wavenumber (very short distances) where it damps the diverging kernel functions. Such damping is important for the FHT which implicitly assumes a band-limited input kernel function. As long as the vertical separation of the source and receiver is quite small relative to the horizontal offset, the vertical offset should not be deleterious. In some cases, the direct integration method of Chave (1983) is able to properly evaluate formally divergent Hankel transform integrals.

There are certainly many other cases where the fields may be calculated correctly, but result in the wrong answer because of numerical instabilities. For most purposes of sources on or below the air/earth interface there should be few problems. It is important to be careful, and to perform simple tests when uncertain abou the calculation. This code has certainly not been optimized for stability, it is only to show the method of calculation. By examining certain scenarios in details, it may be possible to develop more robust codings of the EM field calculation described here.

6 Summary

In this manuscript, we have presented both a description of, and means of calculating, the modal representation of Maxwell's equations for the EM fields in a layered earth. The purpose was to illustrate the underlying simplicity of the complicated stratified earth Green's functions and to present a generalized representation of all EM methods for layered earths. The specific problem of EM sounding was generalized to give the representation arbitrary source/receiver configurations. Two scalar kernel functions

are sufficient to describe the electric kernels of the halfspace sounding problem, a notion consistent with the theoretical development in terms of toroidal and poloidal magnetic modes. A further advantage of the factorization is that it isolates the geometrical portion of the electromagnetic response, facilitating studies of the source-receiver geometry of the observed fields.

References

Backus, G.E., 1986. Poloidal and toroidal fields in geomagnetic field modeling, *Rev. Geophysics*, **24**, 75-109.

Boerner, D.E. & West, G.F., 1989. A generalized representation of the electromagnetic fields in a layered earth, *Geophysical Journal*, **97**, 529–547.

Chave, A.D., 1983. Numerical integration of related Hankel transforms by quadrature and continued fraction expansion, *Geophysics*, **48**, 1671-1686.

Christensen, N.B., 1990. Optimized Fast Hankel Transform Filters, *Geophysical Prospecting*, **38**, 545-568.

Gilbert, F. & Backus, G.E., 1966. Propagator matrices in elastic wave and vibration problems, *Geophysics*, **31**, 326-332.

Johansen, H.K. and Sørensen, K., 1979. Fast Hankel Transforms, *Geophysical Prospecting*, **27**, 876-901.

Kennett, B.L.N., 1983, Seismic Wave Propagation in Stratified Media, Cambridge University Press, Cambridge.

Morse, P.M. & Feshbach, H., 1953, Methods of Theoretical Physics, Parts I and II, McGraw-Hill Inc., New York.

Nobes, D.C., 1984. The inclusion of anisotropy in Maxwell's equations, *Geophys. J. R. astr. Soc.*, **85**, 655-662.

Spies, B.R., and Frischknecht, F., 1991, Electromagnetic Sounding, *in* Electromagnetic Methods in Applied Geophysics - Applications, Part A. *Edited by* Misac N. Nabighian, Society of Exploration Geophysicists, Tulsa, p 285-425..

Ursin, B., 1983. Review of elastic and electromagnetic wave propagation in horizontally layered media, *Geophysics*, **48**, 1063-1081.

Wait, J.R., 1982, Geoelectromagnetism, Academic Press, New York, 268 pages.

Weaver, J.T., 1970. The general theory of electromagnetic induction in a conducting half space, *Geophys. J. R. astr. Soc.*, **22**, 83–100.