Finding the P-wave axes of the Elastic Tensor
Bill Menke, September 26, 2019

Contents:

Section 1: The derivative of wave speed with respect to propagation direction.

The wave polarization direction $\mathbf{p}$ satisfies the eigenvalue problem:

$$M_{ij} p_j = s\, p_i \tag{1}$$

Here $s = \rho v^2$ where ρ is density and v is wave speed. The matrix $\mathbf{M}$ depends upon propagation direction $\mathbf{t}$:

$$M_{ij} = c_{ipjq} t_p t_q \tag{2}$$

Let us now represent the propagation direction $\mathbf{t}$ in terms of polar coordinates θ and φ.

$$\mathbf{t}(\theta,\varphi) = \begin{bmatrix} \sin\theta \sin\varphi \\ \sin\theta \cos\varphi \\ \cos\theta \end{bmatrix} \quad \text{and} \quad \begin{matrix} \theta = \tan\left\{ (t_1^2 + t_2^2)^{1/2} / t_3 \right\} \\ \varphi = \tan2\{t_1, t_2\} \end{matrix} \tag{3}$$

The goal is to compute the derivatives $ds/d\theta$ and $ds/d\varphi$, so that $s(\theta,\varphi)$ can be minimized or maximized with respect to propagation direction.

First-order non-degenerate perturbation theory allows us to calculate the perturbation Δs of an eigenvalue caused by a perturbation $\Delta\mathbf{M}$ of the associated matrix:

$$\Delta s = \Delta M_{ij} p_i p_j \tag{4}$$

I will argue later that non-degenerate perturbation is appropriate in this instance. The derivatives $ds/d\theta$ and $ds/d\varphi$ can be inferred from Equation (4):

$$\Delta s = \frac{ds}{d\theta}\Delta\theta = \frac{dM_{ij}}{d\theta}p_i p_j \Delta\theta \quad \text{so} \quad \frac{ds}{d\theta} = \frac{dM_{ij}}{d\theta}p_i p_j$$

$$\Delta s = \frac{ds}{d\varphi}\Delta\varphi = \frac{dM_{ij}}{d\varphi}p_i p_j \Delta\varphi \quad \text{so} \quad \frac{ds}{d\varphi} = \frac{dM_{ij}}{d\varphi}p_i p_j$$

(5)

Applying the chain rule to the definition of $\mathbf{M}$ in Equation (2) yields:

$$\frac{dM_{ij}}{d\theta} = c_{ipjq}\frac{dt_p}{d\theta}t_q + c_{ipjq}t_p\frac{dt_q}{d\theta} =$$

$$= c_{ipjq}\frac{dt_p}{d\theta}t_q + c_{iqjp}\frac{dt_p}{d\theta}t_q = c_{ipjq}\frac{dt_p}{d\theta}t_q + c_{jpiq}\frac{dt_p}{d\theta}t_q = 2c_{ipjq}\frac{dt_p}{d\theta}t_q$$

$$\frac{dM_{ij}}{d\varphi} = c_{ipjq}\frac{dt_p}{d\varphi}t_q + c_{ipjq}t_p\frac{dt_q}{d\varphi} = 2c_{ipjq}\frac{dt_p}{d\varphi}t_q$$

(6)

Here we have used the fact that $M_{ij} = M_{ji}$ (implying $dM_{ij}/d\theta = dM_{ji}/d\theta$) and $c_{ipjq} = c_{jpiq}$. The derivatives of the propagation direction are computed by differentiating Equation (3):

$$\frac{d\mathbf{t}}{d\theta} = \begin{bmatrix} \cos\theta\sin\varphi \\ \cos\theta\cos\varphi \\ -\sin\theta \end{bmatrix} \quad \text{and} \quad \frac{d\mathbf{t}}{d\varphi} = \begin{bmatrix} \sin\theta\cos\varphi \\ -\sin\theta\sin\varphi \\ 0 \end{bmatrix}$$

Note that $d\mathbf{t}/d\theta$ and $d\mathbf{t}/d\varphi$ are both perpendicular to $\mathbf{t}$.

(7)

Section 2. Gradient method for minimizing/maximizing wave speed.

Actually, we minimize/maximize $s = \rho v^2$.

Step 1. Find an initial guess for (θ, φ) using a coarse grid search (see Part 3).

Step 2: Compute $\mathbf{t}$**,** $d\mathbf{t}/d\theta$ and $d\mathbf{t}/d\varphi$ as in Equations (3) and (7).

Step 3. Compute $\mathbf{M}$, $d\mathbf{M}/d\theta$ and $d\mathbf{M}/d\varphi$ as in Equations (2) and (6).

Step 4. Extract the three eigenvalues λ_i and corresponding eigenvectors $\mathbf{v}^{(i)}$ of $\mathbf{M}$.

Step 5. Find the index of the largest eigenvalue $k = \underset{i}{\operatorname{argmax}}\, \lambda_i$ and set $s = \lambda_k$ and $\mathbf{p} = \mathbf{v}^{(k)}$.

Step 6. Compute the gradient $\mathbf{g} = [ds/d\theta\, , ds/d\varphi]^T$ as in Equation (5) and its direction $\mathbf{n} = \mathbf{g}/|\mathbf{g}|$.

Step 7. Update (θ, φ) using a gradient method, stepping in the either in the $-\mathbf{n}$ or $+\mathbf{n}$ direction, depending upon whether s is being minimized or maximized.

Section 3. Grid search for a starting value.

Step 1. Prepare a coarse grid (θ_m, φ_n) with $0 \leq \theta_m \leq \pi$ and $0 \leq \varphi_n \leq 2\pi$.

Step 2: Then, for each node on the grid, tabulate $s_{mn} = s(\theta_m, \varphi_n)$:

2A. Compute $\mathbf{t}$ as in Equations (3).

2B. Compute $\mathbf{M}$ as in Equations (2).

2C. Extract the three eigenvalues λ_i of $\mathbf{M}$.

2D. Find the index of the largest eigenvalue $k = \max_i \lambda_i$ and set $s_{mn} = \lambda_k$.

Step 3: The starting value (θ_p, φ_q) for minimizing s is:

$$(p, q) = \operatorname*{argmin}_{m,n} s(\theta_m, \varphi_n) \tag{8}$$

The corresponding starting value for maximizing s is:

$$(p, q) = \operatorname*{argmax}_{m,n} s(\theta_m, \varphi_n) \tag{9}$$

The intermediate direction, $\mathbf{t}^{int}$ satisfies $\mathbf{t}^{int} = \pm(\mathbf{t}^{fast} \times \mathbf{t}^{slow})$ where the sign is chosen to insure a right-handed coordinate system $\mathbf{t}^{slow} = \mathbf{t}^{fast} \times \mathbf{t}^{int}$. The intermediate P-wave speed $s^{int} = \lambda_k$ is the largest eigenvalue λ_k of $\mathbf{M}$, were $\mathbf{M}$ is calculated using Equation 2 with $\mathbf{t} = \mathbf{t}^{slow}$. The rotation matrix $\mathbf{S}$ that takes c_{ijpq} into a coordinate system in which $(\mathbf{t}^{fast}, \mathbf{t}^{int}, \mathbf{t}^{slow})$ are parallel to (x_1, x_2, x_3) is $\mathbf{S} = [\mathbf{t}^{fast}, \mathbf{t}^{int}, \mathbf{t}^{slow}]^{\mathrm{T}}$.

Note: I have checked this result numerically and it works fine.

Section 4. I now return to the matter of the appropriateness of applying non-degenerate perturbation theory to the analysis of:

$$\mathbf{M}\mathbf{p}^{(i)} = s_i\, \mathbf{p}^{(i)} \;\rightarrow\; (\mathbf{M} + \Delta\mathbf{M})\left(\mathbf{p}^{(i)} + \Delta\mathbf{p}^{(i)}\right) = (s_i + \Delta s_i)\left(\mathbf{p}^{(i)} + \Delta\mathbf{p}^{(i)}\right) \tag{10}$$

The key question is whether the largest eigenvalue, say s_k is distinct; that is, has a value different than the other two eigenvalues. In typical Earth materials, the answer is yes, since s_k

corresponds to the P-wave velocity, where as the other two eigenvalues refer to the S-wave velocities, and in a typical Earth material the P-velocity is always higher than either of the two S-velocities.

Another interesting aspect of this perturbation problem arises from $d\mathbf{t}/d\theta$ and $d\mathbf{t}/d\varphi$ both being perpendicular to $\mathbf{t}$. This behavior implies $ds/d\theta = ds/d\varphi = 0$ in isotropic material. Denoting $d\mathbf{t}/d\theta = \mathbf{n}$ with $\mathbf{t} \cdot \mathbf{n} = 0$, we find in isotropic material with Lame coefficients λ and μ:

$$c_{ipjq} = \lambda\delta_{ip}\delta_{jq} + \mu\delta_{ij}\delta_{pq} + \mu\delta_{iq}\delta_{jq}$$

$$ds/d\theta = (\lambda\delta_{ip}\delta_{jq} + \mu\delta_{ij}\delta_{pq} + \mu\delta_{iq}\delta_{jp})n_p t_q p_i p_j + (\lambda\delta_{ip}\delta_{jq} + \mu\delta_{ij}\delta_{pq} + \mu\delta_{iq}\delta_{jp})n_p t_q p_i p_j$$
$$= \lambda n_i p_i t_j p_j + \mu n_i t_i p_j p_j + \mu n_j p_j t_i p_i + \lambda n_i p_i t_j p_j + \mu n_p t_p p_i p_i + \mu n_j p_j t_i p_i = 0$$

(11)

Here we have used the fact that, for a P wave in an isotropic material, the polarization direction $\mathbf{p}$ is parallel to the propagation direction $\mathbf{t}$, so $\mathbf{n} \cdot \mathbf{p} = 0$. The same argument applies for $ds/d\varphi$.

Section 5: Discussion of equation for P-wave axes

Suppose that we generically refer to the angles of propagation θ or φ as α. The condition that the wave speed (or rather eigenvalue s) is stationary with respect to small perturbations in α is:

$$0 = \frac{ds}{d\alpha} = \frac{dM_{ij}}{d\alpha} p_i p_j = 2c_{ipjq} \frac{dt_p}{d\alpha} t_q p_i p_j \quad \text{for } \alpha = \theta, \varphi$$

(12)

Defining $b_p \equiv dt_p/d\alpha$ and noting $b_p t_p = 0$, we have

$$0 = (c_{ipjq} p_i p_j) t_q \, b_p \quad \text{for all } \mathbf{b} \perp \mathbf{t}$$

(13)

Consider the eigenvalue problem $N_{pq} t_q = \lambda t_p$ with $N_{pq} = c_{ipjq} p_i p_j$ (where p_j is fixed). Then Equation (13) is equivalent to:

$$0 = \lambda \, t_p \, t_q \, b_p \quad \text{for all } \mathbf{b} \perp \mathbf{t}$$

(14)

Equation (14) is satisfied trivially since $t_p b_p = 0$. Hence the condition for an extremum in s is:

$$c_{ipjq} p_i p_j t_q = \lambda t_p \quad \text{and} \quad c_{ipjq} t_p t_q \, p_j = s \, p_i$$

(15)

After contracting first equation by t_p and the second by p_i :

$$\lambda = c_{ipjq} p_i p_j t_q t_p \quad \text{and} \quad s = c_{ipjq} t_p t_q \, p_i p_j$$

(16)

We conclude $\lambda = s$. We now manipulate Equation (16):

$$c_{ipjq} p_i p_j t_q = s t_p \quad \text{and} \quad c_{ipq} t_p t_q \, p_j = s p_i$$

$$\left(s^{-1} c_{ipjq} p_j t_q\right) p_p = t_i \quad \text{and} \quad \left(s^{-1} c_{ipjq} p_j t_q\right) t_p = p_i$$

$$Z_{ip} \, p_p = t_i \quad \text{and} \quad Z_{ip} \, t_p = p_i \quad \text{with} \quad Z_{ip} \equiv s^{-1} c_{ipjq} p_j t_q$$

(17)

Here the symmetric matrix **Z** both takes **p** into **t** and **t** into **p**. This transformation can happen in either of two ways. The first is when **p**||**t** and $\mathbf{Z} = \mathbf{t}\mathbf{t}^T + \alpha \mathbf{u}\mathbf{u}^T + \beta \mathbf{v}\mathbf{v}^T$, where **u**, **v** and **t** are mutually perpendicular unit vectors and where α and β are constants; that is, **Zy** leaves unchanged the component of **y** parallel to **t** while rescaling the components normal to **t** and/or rotating them in the plane. The second is when $\mathbf{p} \perp \mathbf{t}$ and $\mathbf{Z} = \mathbf{t}\mathbf{p}^T + \mathbf{p}\mathbf{t}^T + \alpha \mathbf{v}\mathbf{v}^T$, where **t**, **p** and **v** are mutually perpendicular unit vectors and where α and β; that is, **Zy** interchanges the **p** and **t** components of **y,** while rescaling the component parallel to **v**. Hence:

$$c_{ipjq} t_p t_j t_q = s t_i \quad \text{with } p_i = t_i \quad \text{or} \quad c_{ipjq} t_p p_j t_q = s p_i \quad \text{with } p_i t_i = 0$$

(18a,b)

Equation (18a) would seem to represent the P-wave and (18b) the S-wave. Unfortunately, I do not know of a fast way of solving Equation (18a). I have, however, checked that it is solved by the $(s, \mathbf{t})$ returned by the linearized solver described above (at least for a test case consisting of arbitrarily rotated c_{ijpq} corresponding to orthorhombic olivine).

```
function [thfast, phfast, sfast, tfast, thint, phint, sint,
tint, thslow, phslow, sslow, tslow, cp] = findaxes2(c)
% find the fast, intermediate and slow directions and rho*v^2 of
P wave in an anisotropic medium
% c: 3x3x3x3 elacticity tensor
% th and ph (in radians) polar angles of axis
% t: unit vector of axi
% s: rho*Vp^2
% cp: c rotated so (fast int slow) are parallel to (x, y, z)

% controls accuracy of gradient method
MAXHALVINGS = 32;
% controls detection of being very close to extermum
MINIMUMLENGTH = 1e-6;

%  PART 1:  Coarse Grid Search
```

```
thmin = 0;
thmax = pi;
phmin = 0;
phmax = 2*pi;
Nth = 19;
Nph = 31;
th = thmin + (thmax-thmin)*[0:Nth-1]'/(Nth-1);
ph = phmin + (phmax-phmin)*[0:Nph-1]'/(Nph-1);
sfast = zeros( Nth, Nph );

for ith=[1:Nth]
for iph=[1:Nph]
    sth = sin(th(ith));
    cth = cos(th(ith));
    sph = sin(ph(iph));
    cph = cos(ph(iph));
    t = [sth*sph; sth*cph; cth];
    % I checked that t'*dtdth=0 and t'*dtdph=0
    M = zeros(3,3);
    dMdth = zeros(3,3);
    dMdph = zeros(3,3);
    for i=[1:3]
    for j=[1:3]
    for p=[1:3]
    for q=[1:3]
        M(i,j) = M(i,j) + c(i,p,j,q)*t(p)*t(q);
    end
    end
    end
    end
    [V,L] = eig(M,'vector');
    sfast( ith, iph ) = max(L);
end
end

[s1,k1] = max(sfast);
[s2,k2] = max(s1);
k3 = k1(k2);
ithmax = k3;
iphmax = k2;
sgridmax = sfast(ithmax,iphmax);
thmax = th(ithmax);
phmax = ph(iphmax);

[s1,k1] = min(sfast);
[s2,k2] = min(s1);
```

```matlab
k3 = k1(k2);
ithmin = k3;
iphmin = k2;
sgridmin = sfast(ithmin,iphmin);
thmin = th(ithmin);
phmin = ph(iphmin);

% Part 2, refine fast axis

myth = thmax;
myph = phmax;
alpha = (pi/180) * 1;
halvings = 0;

for itt=[1:100]
sth = sin(myth);
cth = cos(myth);
sph = sin(myph);
cph = cos(myph);

t = [sth*sph; sth*cph; cth];
dtdth = [cth*sph; cth*cph; -sth];
dtdph = [sth*cph; -sth*sph; 0];
M = zeros(3,3);
dMdth = zeros(3,3);
dMdph = zeros(3,3);
for i=[1:3]
for j=[1:3]
for p=[1:3]
for q=[1:3]
    M(i,j) = M(i,j) + c(i,p,j,q)*t(p)*t(q);
    dMdth(i,j) = dMdth(i,j) + 2*c(i,p,j,q)*dtdth(p)*t(q);
    dMdph(i,j) = dMdph(i,j) + 2*c(i,p,j,q)*dtdph(p)*t(q);
end
end
end
end
[V,L] = eig(M,'vector');
[Lmax, k] = max(L);
mys = Lmax;
P = V(:,k);
mydsdth = 0;
mydsdph = 0;
for i=[1:3]
for j=[1:3]
    mydsdth = mydsdth + dMdth(i,j)*P(i)*P(j);
    mydsdph = mydsdph + dMdph(i,j)*P(i)*P(j);
```

```
end
end

grad_s = [mydsdth; mydsdph];
nu = grad_s/sqrt(grad_s'*grad_s);

myth2 = myth + alpha * nu(1);
myph2 = myph + alpha * nu(2);

sth2 = sin(myth2);
cth2 = cos(myth2);
sph2 = sin(myph2);
cph2 = cos(myph2);

t2 = [sth2*sph2; sth2*cph2; cth2];
M2 = zeros(3,3);

for i=[1:3]
for j=[1:3]
for p=[1:3]
for q=[1:3]
    M2(i,j) = M2(i,j) + c(i,p,j,q)*t2(p)*t2(q);
end
end
end
end
[V2,L2] = eig(M2,'vector');
[L2max, k2] = max(L2);
mys2 = L2max;
if( mys2 > mys )
    myth = myth2;
    myph = myph2;
    mys = mys2;
else
    alpha = alpha/2;
    halvings = halvings + 1;
end
if( halvings > MAXHALVINGS )
    break;
end

end
thfast = myth;
phfast = myph;
sfast = mys;

% Part 3, refine slow axis
```

```
myth = thmin;
myph = phmin;
alpha = (pi/180) * 1;
halvings = 0;

for itt=[1:100]
sth = sin(myth);
cth = cos(myth);
sph = sin(myph);
cph = cos(myph);

t = [sth*sph; sth*cph; cth];
dtdth = [cth*sph; cth*cph; -sth];
dtdph = [sth*cph; -sth*sph; 0];
M = zeros(3,3);
dMdth = zeros(3,3);
dMdph = zeros(3,3);
for i=[1:3]
for j=[1:3]
for p=[1:3]
for q=[1:3]
    M(i,j) = M(i,j) + c(i,p,j,q)*t(p)*t(q);
    dMdth(i,j) = dMdth(i,j) + 2*c(i,p,j,q)*dtdth(p)*t(q);
    dMdph(i,j) = dMdph(i,j) + 2*c(i,p,j,q)*dtdph(p)*t(q);
end
end
end
end
[V,L] = eig(M,'vector');
[Lmin, k] = max(L); % code path min() -> max(), Menke 02/11/20
mys = Lmin;
P = V(:,k);
mydsdth = 0;
mydsdph = 0;
for i=[1:3]
for j=[1:3]
    mydsdth = mydsdth + dMdth(i,j)*P(i)*P(j);
    mydsdph = mydsdph + dMdph(i,j)*P(i)*P(j);
end
end

grad_s = [mydsdth; mydsdph];
len_grad_s = sqrt(grad_s'*grad_s);
if (len_grad_s < MINIMUMLENGTH )
    break;
end
```

```
nu = -grad_s/len_grad_s;

myth2 = myth + alpha * nu(1);
myph2 = myph + alpha * nu(2);

sth2 = sin(myth2);
cth2 = cos(myth2);
sph2 = sin(myph2);
cph2 = cos(myph2);

t2 = [sth2*sph2; sth2*cph2; cth2];
M2 = zeros(3,3);

for i=[1:3]
for j=[1:3]
for p=[1:3]
for q=[1:3]
    M2(i,j) = M2(i,j) + c(i,p,j,q)*t2(p)*t2(q);
end
end
end
end
[V2,L2] = eig(M2,'vector');
[L2min, k2] = max(L2);
mys2 = L2min;
if( mys2 < mys )
    myth = myth2;
    myph = myph2;
    mys = mys2;
else
    alpha = alpha/2;
    halvings = halvings + 1;
end
if( halvings > MAXHALVINGS )
    break;
end

end
thslow = myth;
phslow = myph;
sslow = mys;

% Part 4, intermediate axis, perpendicular to other axes
sth = sin(thfast);
cth = cos(thfast);
sph = sin(phfast);
cph = cos(phfast);
```

```
tfast = [sth*sph; sth*cph; cth];

sth = sin(thslow);
cth = cos(thslow);
sph = sin(phslow);
cph = cos(phslow);
tslow = [sth*sph; sth*cph; cth];

tint = cross(tfast, tslow);
thint = atan( sqrt(tint(1)*tint(1)+tint(2)*tint(2)) / tint(3) );
phint = atan2( tint(1), tint(2) );

sth = sin(thint);
cth = cos(thint);
sph = sin(phint);
cph = cos(phint);
tint = [sth*sph; sth*cph; cth];

% ensure sign correct; that is fast cross intermediate = slow
if( tslow'*cross(tfast,tint) < 0 )
    tint = -tint;
end

thint = atan( sqrt(tint(1)*tint(1)+tint(2)*tint(2)) / tint(3) );
phint = atan2( tint(1), tint(2) );
% I check that [tint'*tfast, tint'*tslow, tfast'*tslow ]=[0,0,0]

M = zeros(3,3);
for i=[1:3]
for j=[1:3]
for p=[1:3]
for q=[1:3]
    M(i,j) = M(i,j) + c(i,p,j,q)*tint(p)*tint(q);
end
end
end
end
L = eig(M);
sint = max(L);

% rotate to these axes
cp = rot3x3x3x3( c, [tfast, tint, tslow]' );

end
function [Cout] = rot3x3x3x3(Cin,S)
Cout = zeros(3,3,3,3);
```

```
for i=[1:3]
for j=[1:3]
for k=[1:3]
for l=[1:3]
    for p=[1:3]
    for q=[1:3]
    for r=[1:3]
    for s=[1:3]
        Cout(i,j,k,l) = Cout(i,j,k,l) +
S(i,p)*S(j,q)*S(k,r)*S(l,s)*Cin(p,q,r,s);
    end
    end
    end
    end
end
end
end
end
end
```