Isotropic and Vertical Compensated Linear Vector Dipole Component of Moment Tensor Trade Off in Inversions that use Surface Wave Data

William Menke, June 26, 2024

Consider a centroid moment tensor (CMT) inversions that relies heavily on Love and Rayleigh waves (which typically are the highest amplitude parts of a teleseismic seismogram). Now suppose two cases:

The moment tensor consisting of an earthquake part plus an isotropic (ISO) part $M = M_{eq} + \text{diag}(A, A, A)$;

and

The moment tensor consisting of an earthquake part plus a Vertical Compensated Linear Vector Dipole (VCLVD) part $M = M_{eq} + \text{diag}(B, B, -2B)$.

Aki and Richards (2009) give the following Love wave excitation:

$$u_x^{\text{LOVE}}(\mathbf{x}, \omega) = -\sum_n \sin\phi \frac{l_1(z)}{8cUI_1}\sqrt{\frac{2}{\pi k_n r}} \exp\left[i\left(k_n r + \frac{\pi}{4}\right)\right]$$
$$\times \left\{ ik_n l_1(h)[M_{xx}\sin\phi\cos\phi - M_{yx}\cos^2\phi + M_{xy}\sin^2\phi \quad (7.148)\right.$$
$$\left. - M_{yy}\sin\phi\cos\phi] - \frac{dl_1}{dz}\bigg|_h [M_{xz}\sin\phi - M_{yz}\cos\phi] \right\}.$$

The sum is over Love wave modes, but in this analysis, I consider only the fundamental mode mode $n = 0$ (which typically has the highest amplitude). The Love wave only depends on $(M_{xx},\ M_{yy},\ M_{zz})$ through the term $(M_{xx} - M_{yy})\sin\varphi\sin\varphi$ term, which cancels in both the ISO and VCLVD cases. Thus, the Love wave cannot detect either.

Aki and Richards (2009) give the following Rayleigh wave excitation:

$$u_z^{\text{RAYLEIGH}} = \sum_n \frac{r_2(z)}{8cUI_1}\sqrt{\frac{2}{\pi k_n r}}\exp\left[i\left(k_n r + \frac{\pi}{4}\right)\right] \times \left\{k_n r_1(h)[M_{xx}\cos^2\phi + (M_{xy}+M_{yx})\sin\phi\cos\phi + M_{yy}\sin^2\phi] + i\left.\frac{dr_1}{dz}\right|_h [M_{xz}\cos\phi + M_{yz}\sin\phi] - ik_n r_2(h)[M_{zx}\cos\phi + M_{zy}\sin\phi] + \left.\frac{dr_2}{dz}\right|_h M_{zz}\right\}. \quad (7.150)$$

Again, I consider only the fundamental mode $n = 0$. The two cases have the same excitation when

$$c_1(A\cos^2\theta + A\sin^2\theta) + c_2 A = c_1(B\cos^2\theta + B\sin^2\theta) - 2c_2 B$$

Here the $c_1 \equiv k_0 r_1$ and $c_2 \equiv dr_2/dz$ are frequency-dependent functions and θ is source-receiver azimuth. The two cases are equal when

$$B = \frac{(c_1 + c_2)}{(c_1 - 2c_2)}A$$

Hence, at a single frequency, ISO and CLVD exactly trade off, and do so without affecting the earthquake part of the moment tensor. A CMT inversion that imposes the constraint $M_{xx} + M_{yy} + M_{zz} = 0$ will lead to $A = 0$ and $B \neq 0$, without changing the earthquake part of the moment tensor.

When the moment tensor is considered frequency-independent, this trade off can be accomplished only at a specific frequency, because the c's are frequency-dependent functions). However, as CMT inversions typically are fairly narrow band, ISO and CLVD will approximately trade off, as is shown in the test inversion (Fig 1). This inversion uses a Rayleigh wave in a half space, a 60° normal fault with 10% ISO superimposed. The $M_{xx} + M_{yy} + M_{zz} = 0$ constraint leads to zero ISO and non-zero CLVD, and about 1% change in $\mathbf{M}_{eq}$:

$$\mathbf{M}^{true} = \begin{bmatrix} 0.866 & 0.000 & -0.500 \\ 0.000 & 0.000 & 0.000 \\ -0.500 & 0.000 & -0.866 \end{bmatrix} + \begin{bmatrix} 0.100 & 0.000 & 0.000 \\ 0.000 & 0.100 & 0.000 \\ 0.000 & 0.000 & 0.100 \end{bmatrix}$$

$$\mathbf{M}^{est} = \begin{bmatrix} 0.872 & 0.000 & -0.500 \\ 0.000 & 0.000 & 0.000 \\ -0.500 & 0.000 & -0.872 \end{bmatrix} + \begin{bmatrix} 0.015 & 0.000 & 0.000 \\ 0.000 & 0.015 & 0.000 \\ 0.000 & 0.000 & -0.030 \end{bmatrix}$$

Note that the estimated CLVD component is quite a bit smaller than the true ISO component.

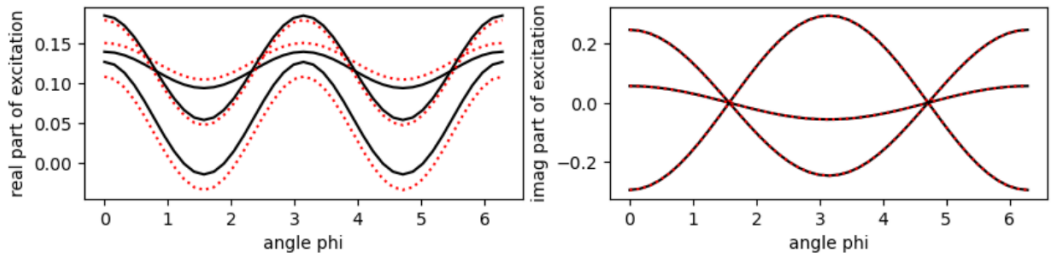

Fig 1. Test CMT inversion. (Left) Real part of true (black) and predicted (red dotted) Rayleigh wave excitation for three frequencies spanning a factor of ten in size. (Right) Same a left, but for imaginary part.

When the moment tensor is considered frequency-dependent, this trade off can be accomplished only at all frequencies, with the frequency dependence of the ISO and CLVD component being different from one another.

Reference

Aki, K. and P.G. Richards, Quantitative Seismology, Second Edition, University Science Books, Mill Valley, California (USA), 743 pp, 2009.

Python Code

```
# theta-axis
Lth = 41;
thmin = -pi;
thmax = pi;
Dth = (thmax-thmin)/(Lth-1);
th = gda_cvec(Dth*np.linspace(0,Lth-1,Lth));

# material properties
r = sqrt(3.0);
beta = 4.0;
alpha = r*beta;
cr = 0.91*beta;

# vertical wavefunctions, A&R 2009 p326
def r1r2(k, z):
    kz = k*z;
    a0 = -0.8475;
    c0 = -0.3933;
    b11 = 1.0;
    b12 = -0.5773;
    b21 =  0.8475;
    b22 = -1.4679;
    r1 = b11*exp(a0*kz) + b12*exp(c0*kz);
    r2 = b21*exp(a0*kz) + b22*exp(c0*kz);
    dr1dz = a0*b11*exp(a0*kz) + c0*b12*exp(c0*kz);
    dr2dz = a0*b21*exp(a0*kz) + c0*b22*exp(c0*kz);
    return r1, r2, dr1dz, dr2dz;
```

```
# unnormalized excittion, A&R 2009 p326
def excit( M, k, h, th ):
    cth = cos(th);
    sth = sin(th);
    cth2 = cth**2;
    sth2 = sth**2;
    sthcth = sth*cth;
    r1h, r2h, dr1dzh, dr2dzh = r1r2(k, h);
    t1R =  k*r1h*( M[0,0]*cth2 + (M[0,1]+M[1,0])*sthcth + M[1,1]*sth2 );
    t2I = dr1dzh*( M[0,2]*cth  + M[1,2]*sth );
    t3I = -k*r2h*( M[2,0]*cth  + M[2,1]*sth );
    t4R = dr2dzh*M[2,2];
    return t1R+t4R, t2I+t3I;

ISNORMAL=1;

# urnormalized moment tensor,,A&R 2009 p51
def moment( NORM, vol ):
    if( NORM ):
        ph = 60.0*pi/180.0;
        sph = sin(ph);
        cph = cos(ph)
        n = np.array( [ [sph], [0.0],  [cph] ] );
        s = np.array( [ [cph], [0.0],  [-sph] ] );
    else:
        ph = 30.0*pi/180.0;
        sph = sin(ph);
        cph = cos(ph)
        n = np.array( [ [sph],  [0.0],  [cph] ] );
        s = np.array( [ [-cph], [0.0],  [sph] ] );
    Mxyz = np.matmul(n,s.T)+np.matmul(s,n.T);
    Mxyz = Mxyz + vol*np.identity(3);
    return(Mxyz);

vol=0.1;
M = moment( ISNORMAL, vol );
print(M);
print(M-np.trace(M)*np.identity(3)/3);
eval, evec = la.eigh(M);
ISO = (eval[2]+eval[1]+eval[0])/3.0;
DC = (eval[2]-eval[0]-abs(eval[2]+eval[0]-2.0*eval[1]))/2.0;
CLVD = 2.0*(eval[2]+eval[0]-2.0*eval[1])/3.0;
print("ISO %.2f DC %.2f CLVD %.2f" % (ISO,DC,CLVD) );
htrue = 0.5;

wlist = np.array( [0.5, 2.0, 5.0 ] );
Nw, = np.shape(wlist);

dR = np.zeros((Lth,Nw));
dI = np.zeros((Lth,Nw));
dtrue = np.zeros((2*Nw*Lth,1));
for iw in range(Nw):
    k = wlist[iw]/cr; # cr = w/k
```

```
    for ith in range(Lth):
        th0 = th[ith,0];
        tR, tI = excit( M, k, htrue, th0);
        dR[ith,iw] = tR;
        dI[ith,iw] = tI;
        dtrue[2*iw*Lth+ith,0]=tR;
        dtrue[2*iw*Lth+Lth+ith,0]=tI;
dobs = np.copy(dtrue);

# data kernel
htry = 1.00*htrue;
G = np.zeros((2*Lth*Nw,6));
ilist = np.array( [0, 1, 2, 0, 0, 1] );
jlist = np.array( [0, 1, 2, 1, 2, 2] );
for q in range(6):
    i0 = ilist[q];
    j0 = jlist[q];
    Mr = np.zeros((3,3));
    Mr[i0,j0]=1.0;
    Mr[j0,i0]=1.0;
    for iw in range(Nw):
        k = wlist[iw]/cr; # cr = w/k
        for ith in range(Lth):
            th0 = th[ith,0];
            tR, tI = excit(Mr, k, htry, th0);
            G[2*iw*Lth+ith,q] = tR;
            G[2*iw*Lth+Lth+ith,q] = tI;

HNODC = 1.0e3*np.array( [ [1.0, 1.0, 1.0, 0.0, 0.0, 0.0] ] );
hnodc = 1.0e3*np.zeros((1,1));

HSMALL = 1.0e-4*np.identity(6);
hsmall = 1.0e-4*np.zeros((6,1));

F = np.concatenate( (G,HNODC,HSMALL), axis=0 );
f = np.concatenate( (dobs,hnodc,hsmall), axis=0 );

# least quares solution
mest = la.solve( np.matmul(F.T,F), np.matmul(F.T,f) );
Mest = np.zeros((3,3));
for q in range(6):
    i0 = ilist[q];
    j0 = jlist[q];
    Mest[i0,j0]=mest[q,0];
    Mest[j0,i0]=mest[q,0];
dpre = np.matmul(G,mest);

dRpre = np.zeros((Lth,Nw));
dIpre = np.zeros((Lth,Nw));
for iw in range(Nw):
    for ith in range(Lth):
        dRpre[ith,iw] = dpre[2*iw*Lth+ith,0];
        dIpre[ith,iw] = dpre[2*iw*Lth+Lth+ith,0];
```

```
eval, evec = la.eigh(Mest);
ISO = (eval[2]+eval[1]+eval[0])/3.0;
DC = (eval[2]-eval[0]-abs(eval[2]+eval[0]-2.0*eval[1]))/2.0;
CLVD = 2.0*(eval[2]+eval[0]-2.0*eval[1])/3.0;
print(Mest);
print(Mest+np.diag([0.5*CLVD,0.5*CLVD,-CLVD]));
print("ISO %.2f DC %.2f CLVD %.2f" % (ISO,DC,CLVD) );
e = dobs-dpre;
rms = sqrt( np.matmul(e.T,e)[0,0]/(2.0*Lth) );
print("rms error %f" % (rms) );

fig1 = plt.figure(figsize=(10,2));
ax1 = plt.subplot(1,2,1);
plt.xlabel('angle phi');
plt.ylabel('real part of excitation');
for iw in range(Nw):
    plt.plot(th, dR[0:Lth,iw:iw+1], 'k-' );
    plt.plot(th, dRpre[0:Lth,iw:iw+1], 'r:' );
ax2 = plt.subplot(1,2,2);
plt.xlabel('angle phi');
plt.ylabel('imag part of excitation');
for iw in range(Nw):
    plt.plot(th, dI[0:Lth,iw:iw+1], 'k-' );
    plt.plot(th, dIpre[0:Lth,iw:iw+1], 'r:' );
plt.show();
```