Error in S wave polarization angle due to the free surface

Bill Menke, November 15, 2013

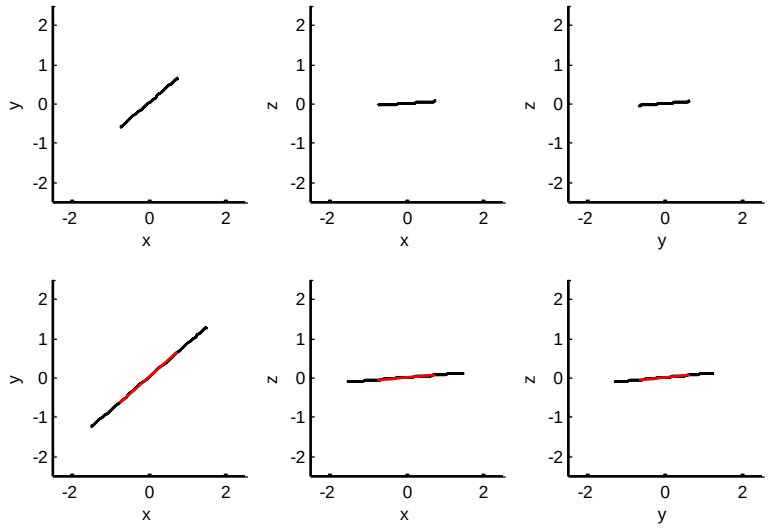
As is pointed out in Aki and Richards, Quantitative Seismology, Vol. 1, 1980, Page 190, Problem 5.6, the displacement **U** of the free surface is not the same as the polarization **P** of an incident S wave because of the interfering effect of the reflected S wave and converted P wave.

The polarization direction **P** of the incident S wave depends on two parameters, the angle of incidence j of the S wave and the relative amount of SV and SH motion (which I parameterize by the angle q from the SV direction).

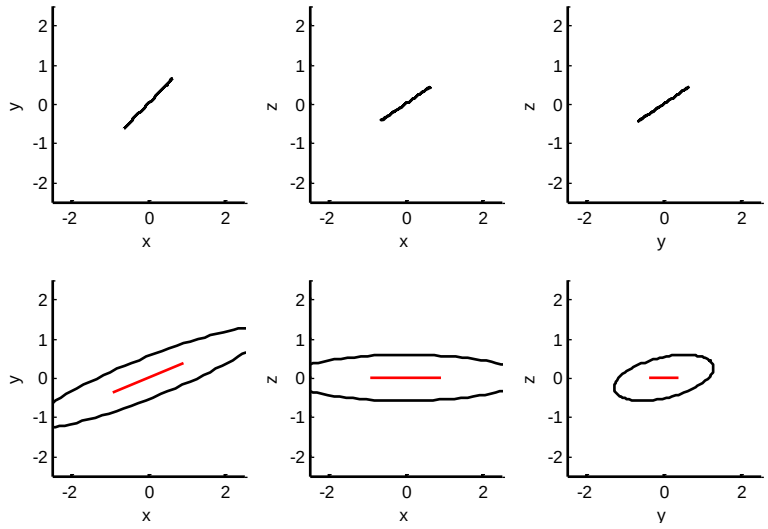
I study here the case of a half space with Vp=6.5 km/s and Vp/Vs=1.78. A key quantity is the critical angle $\cos^{-1}$(Vs/Vp), which is 33.749 deg in this case. At angles of incidence steeper than the critical angle, the displacement **U** of the free surface is linear, but is misaligned with the polarization direction **P**. At angles of incidence more shallow than the critical angle, the P wave is evanescent and the motion of the free surface is elliptical.

A linear motion is easy to compare with another linear motion; the angle between them is a measure of the error E in misalignment. An elliptical motion is harder to compare with a linear motion. I use eigenvalue analysis to extract the orientation of the ellipse, and use its longest axis as a proxy for its orientation. I start with cosine wave for the incident S wave and use Aki and Richard's formulas to compute time series $U_i(t)$. They are sinusoidal, but in general are phase shifted with respect to a cosine. I then form the matrix $M_{ij}=U_i(t)\star U_j(t)$, where $\star$ denotes cross-correlation. The eigenvector **u** corresponding to the largest eigenvalue is used as a proxy for the "average" orientation of **U**. This makes most sense when the ellipse is actually elongated.

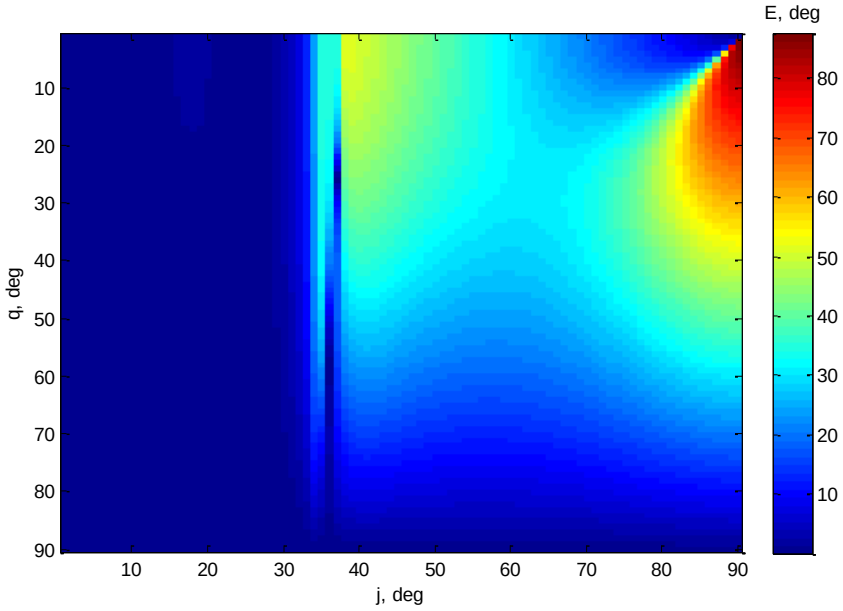
The figure below shows the plots of **P** (top row) and **U** and **u** (bottom row, with **U** black and **u** red)in black, for the case q=40, j=4.



This is a steep angle of incidence case, and **P** and **U** are nearly parallel and the error is small (E=0.3 deg). On the other hand, the q=40, j=34 case corresponds to an angle of incidence just a little larger than the critical angle:



In this case the error is large (33.8 deg). The Matlab script that produced these plots is shown below. The following figure shows the error E(i, q) as a function of angle of incidence j and amount of SH motion q. Note that the error is relatively small (<10 deg) when the angle of incidence j is less that the critical angle (about 34 deg) and when the motion is close to being purely SH (q>75 deg). (SH waves generate no converted P waves, so the free surface motion **U** is always horizontal and parallel to **P** in the pure SH case).



```
clear all;

a = 6.5;
b = a/1.8;

% orientation of incident S wave
q = 40;
ASV = cos( (pi/180) * q );
ASH = sin( (pi/180) * q );

for tj=[33:1:89]

% tj is angle of incidence of incident and reflected S wave
tjc = (180/pi)*asin(b/a); % critical S wave angle
stj = sin( (pi/180)*tj ); % sin of tj
ctj = cos( (pi/180)*tj ); % cos of tj
ti = (180/pi) * asin( (a/b) * stj ); % angle of incidence of converted P wav
sti = sin( (pi/180)*ti ); % sin of ti
cti = cos( (pi/180)*ti ); % cos of ti

% polarization vector of incident S wave
Px = ASV * ctj;
Py = ASH;
Pz = ASV * stj;

% cti should either be +real (downward propagating wave)
% or                   +imag (decaying with depth)
if( tj < tjc )
    if( real(cti) < 0 )
        cti = (-cti);
    end
end
if( tj > tjc )
    if( imag(cti) < 0 )
        cti = (-cti);
    end
end

% free surface displacement
% see Aki and Richards, Quantitative Seismology, Vol 1, Page 190
% Problem 5.6.
p = stj/b;  % horizontal slowness
ctioa = cti/a; ctjob = ctj/b; % vertical slownesses
x = ( (1/(b*b))-(2*p*p) );
D = (x*x) + (4*p*p*ctioa*ctjob);
Ux = (ASV*(2/b)*ctjob*x) / D;
Uy = ASH*2;
Uz = (ASV*(4*p/b)*ctioa*ctjob) / D;

% Aki and Richards use the phase factor exp( iw(px-t) )
% which for x=0 reduces to exp(-iwt). Suppose the incident
% wave has a cosine time dependence of real amplitude A.
% We create it by summing positive and negative frequencies,
% A* exp(+iwt) + A exp(-iwt) = 2A cos(wt).  So the free
% surface displacement has a time dependence
```

```
% B* exp(+iwt) + B exp(-iwt), which is real has both sin(wt)
% and cos(wt) terms, and is thus phase lagges with respect
% to the incident wave.

N = 100;
t = (2*pi/(N-1))*[0:N-1]';
w = 1;

Prx = Px*cos(w*t);
Pry = Py*cos(w*t);
Prz = Pz*cos(w*t);

Urx = (conj(Ux)*exp(complex(0,1)*w*t) + Ux*exp(complex(0,-1)*w*t))/2;
Ury = (conj(Uy)*exp(complex(0,1)*w*t) + Uy*exp(complex(0,-1)*w*t))/2;
Urz = (conj(Uz)*exp(complex(0,1)*w*t) + Uz*exp(complex(0,-1)*w*t))/2;
Urx=real(Urx); Ury=real(Ury); Urz=real(Urz);

figure(1);
clf;
subplot(2,3,1);
set(gca, 'LineWidth', 2);
hold on;
axis( [-2.5, 2.5, -2.5, 2.5] );
plot( Prx, Pry, 'k-', 'LineWidth', 2 );
xlabel('x');
ylabel('y');

subplot(2,3,4);
set(gca, 'LineWidth', 2);
hold on;
axis( [-2.5, 2.5, -2.5, 2.5] );
plot( Urx, Ury, 'k-', 'LineWidth', 2 );
xlabel('x');
ylabel('y');

subplot(2,3,2);
set(gca, 'LineWidth', 2);
hold on;
axis( [-2.5, 2.5, -2.5, 2.5] );
plot( Prx, Prz, 'k-', 'LineWidth', 2 );
xlabel('x');
ylabel('z');

subplot(2,3,5);
set(gca, 'LineWidth', 2);
hold on;
axis( [-2.5, 2.5, -2.5, 2.5] );
plot( Urx, Urz, 'k-', 'LineWidth', 2 );
xlabel('x');
ylabel('z');

subplot(2,3,3);
set(gca, 'LineWidth', 2);
hold on;
axis( [-2.5, 2.5, -2.5, 2.5] );
```

```
plot( Pry, Prz, 'k-', 'LineWidth', 2 );
xlabel('y');
ylabel('z');

subplot(2,3,6);
set(gca, 'LineWidth', 2);
hold on;
axis( [-2.5, 2.5, -2.5, 2.5] );
plot( Ury, Urz, 'k-', 'LineWidth', 2 );
xlabel('y');
ylabel('z');

M = [Urx, Ury, Urz]' * [Urx, Ury, Urz];
[V,L] = eig(M);
[Lmax, iLmax] = max(diag(L));
Uax = V(1,iLmax); Uay = V(2,iLmax); Uaz = V(3,iLmax);

subplot(2,3,4);
plot( [-Uax, Uax], [-Uay, Uay], 'r-', 'LineWidth', 2 );

subplot(2,3,5);
plot( [-Uax, Uax], [-Uaz, Uaz], 'r-', 'LineWidth', 2 );

subplot(2,3,6);
plot( [-Uay, Uay], [-Uaz, Uaz], 'r-', 'LineWidth', 2 );


LP = sqrt(Px*Px + Py*Py + Pz*Pz);
LU = sqrt(Uax*Uax + Uay*Uay + Uaz*Uaz);
PdotU = (Px*Uax+Py*Uay+Pz*Uaz);
E = (180/pi) * acos( PdotU / (LP*LU) );
if( E>90)
    E = E-180;
end
if( E<(-90) );
    E = E+180;
end
fprintf('E(%d): %f\n', tj, E );

zzz = ginput (1);

end
```