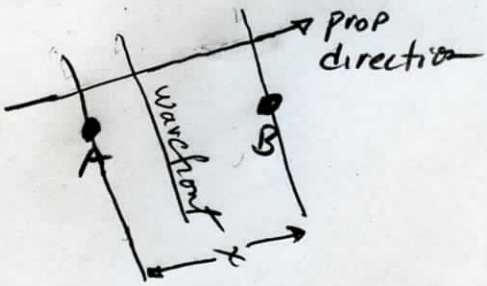
Dec 3, 2013

Determining phase velocity of surface waves by waveform fitting



① Assume

$$B(t) = A(t) * f(t)$$

$f(t)$ = phase-shift filter

$f(\omega) = \exp(-i\omega x / v(\omega))$

$v(\omega)$ = phase velocity.

② note $|A(\omega) f(\omega)|^2 = (A^*A)\, e^{+i\omega x/v}\, e^{-i\omega x/v} = A^*A$

so phase shift filter doesn't change spectral power, or by Parseval's Thm, $\int A^2(t)\,dt$ = energy = E_A

③ minimizing $\xi = \int [A*f - B]^2\,dt$ L_2 norm of misfit with respect to $v(\omega)$ The same as maximizing zero lag cross correlation $(A*B)(t=0)$ since

$$E_{AB} = \int A^2(t)\,dt + \int B^2(t)\,dt - 2\int A(t)\,B(t)\,dt$$
$$= E_A + E_B + (A*B)(t=0)$$

since E_A and E_B not a fcn of $v(\omega)$.

④ Strategy: 1. represent $v(\omega)$ as set of control points $\underline{v}$ in some spline (eg linear spline)

2. compute $\partial E_{AB} / \partial v_i$ by finite differences

3. use gradient method to minimize ξ wrt $\underline{v}$

⑤ symmetry considerations:

note $v(-f) = v(f)$

```
vpos = interp1(f, v, fpos, 'linear', 'extrap');
vfull = [vpos', fliplr(vpos(2:Nf-1)')]';
```

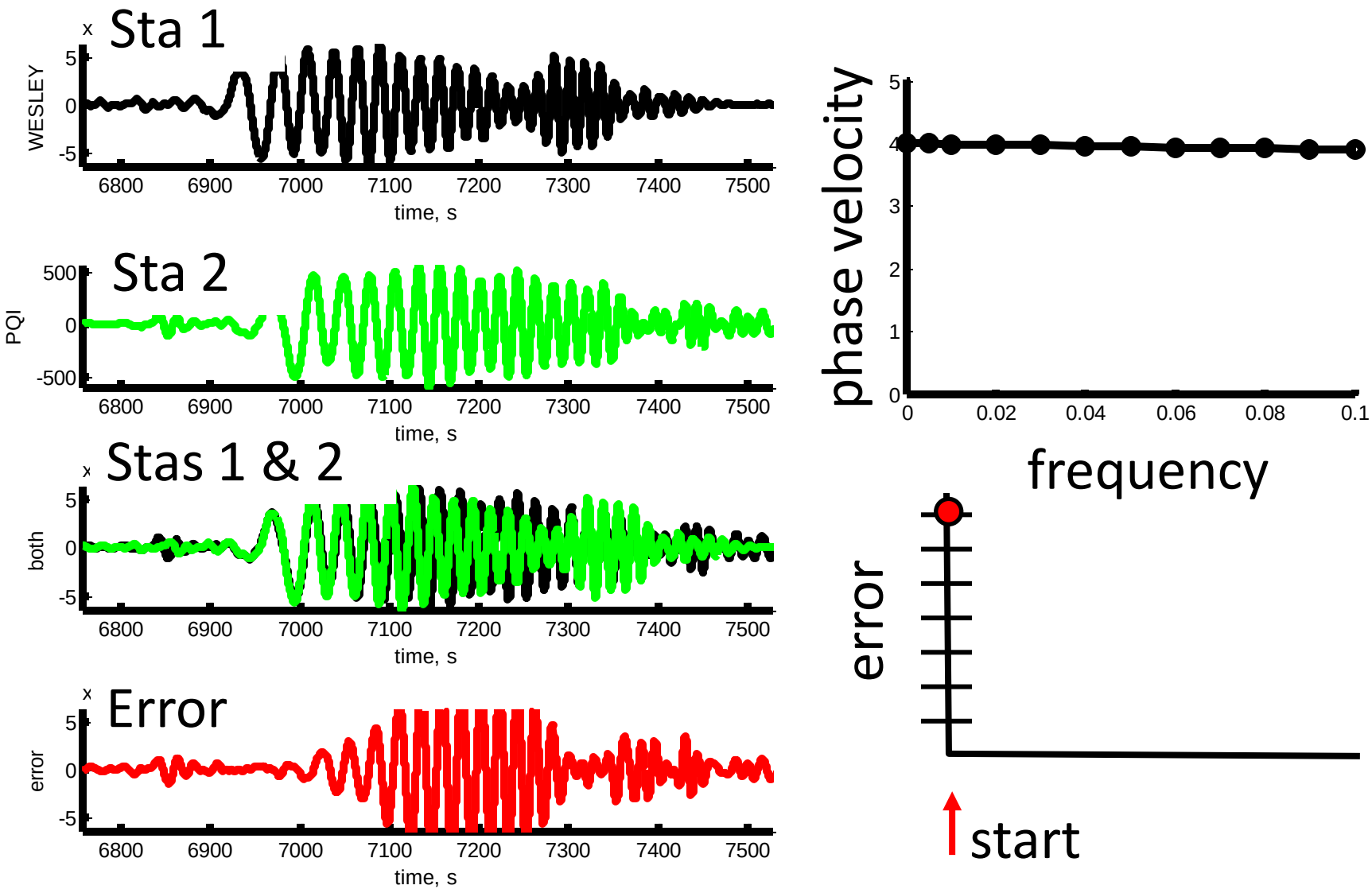
Example: Steps 1, 2, 3: Hand-fitting of dispersion fcn

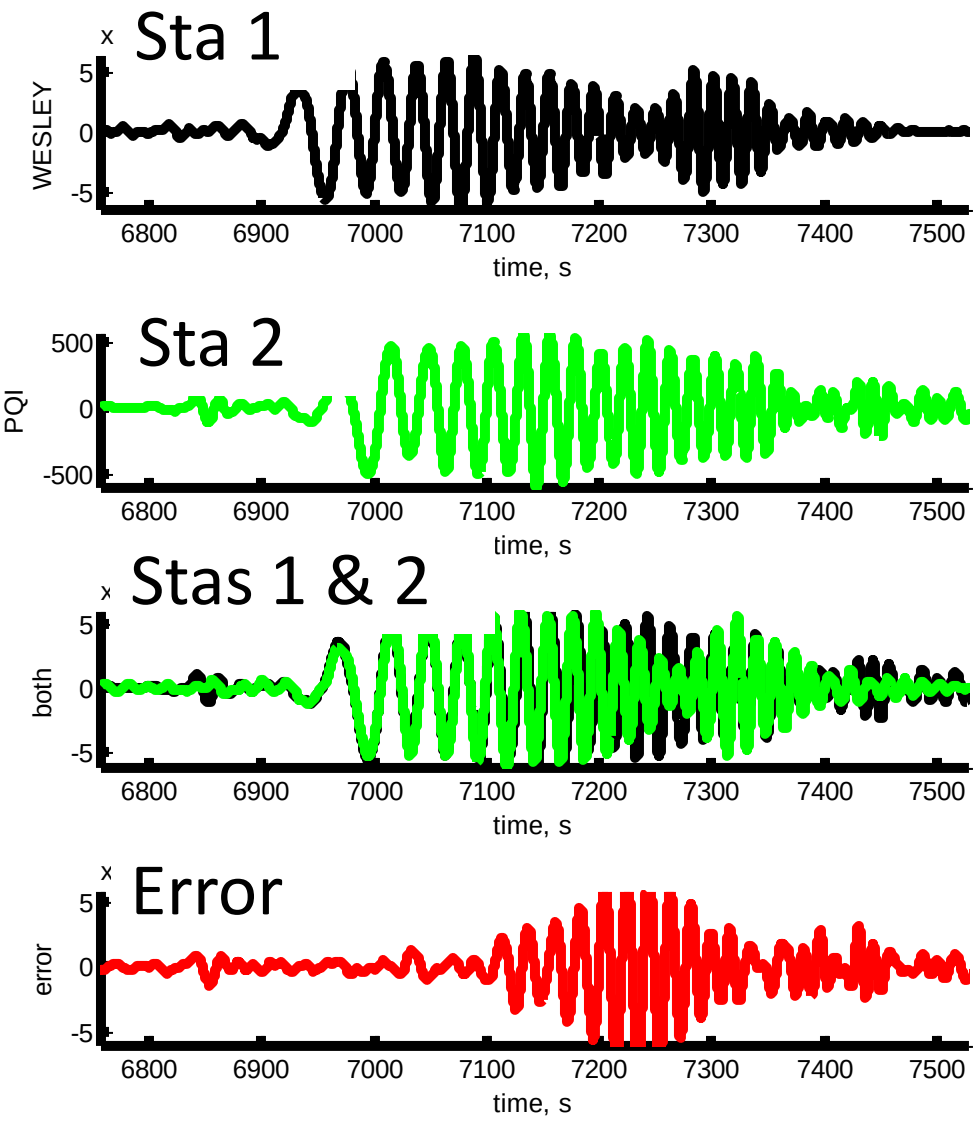
```
% dispersion diagram information
if( exist('dispersion.txt', 'file') )
    DP = load('dispersion.txt');
    fx = DP(:,1);
    vx = DP(:,2);
else
    fx = [ 0.000, 0.005, 0.010, 0.020, 0.030, 0.040, 0.050, 0.060,
0.070, 0.080, 0.090, 0.100, fmax ]';
    vx = [ 4.000, 3.990, 3.980, 3.970, 3.960, 3.950, 3.940, 3.930,
3.920, 3.910, 3.900, 3.890, 3.500 ]';
end
Nx = length(fx);
vpos = interp1(fx,vx,fpos,'linear','extrap');
vfull = [vpos', fliplr( vpos(2:Nf-1)' ) ]';

…

% linear coefficient of proportionality
c = max(abs(dsta1)) / max(abs(dsta2));

…

% perturb dispersion
[fn, vn] = ginput(1);
if( vn<1)
    break
end
ef = (fx-fn).^2;
[efmin, j] = min(ef);
vx(j) = vn;
Nx = length(fx);
vpos = interp1(fx,vx,fpos,'linear','extrap');
vfull = [vpos', fliplr( vpos(2:Nf-1)' ) ]';

…

% advance phase of station 1
dtildea = fft(dsta1);
x = rcsta1 - rcsta2;
dtildead = dtildea .* exp( i*x*w./vfull );
da = real(ifft( dtildead ));

…

e = c*dsta2 - da;
E = e'*e;
```

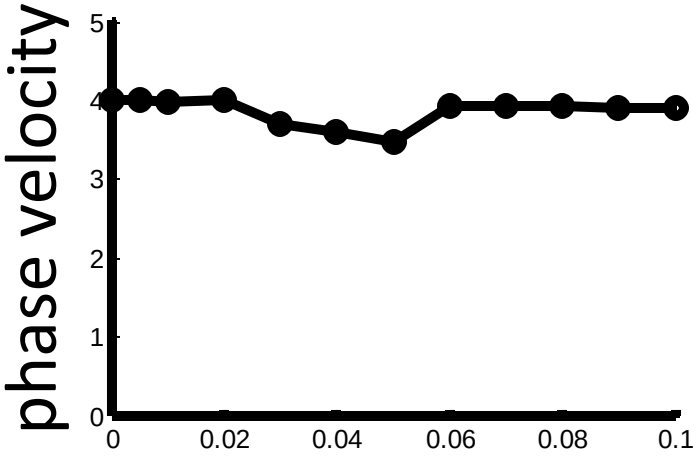
Example: Step 4: Gradient Method to fine-tune dispersion fcn

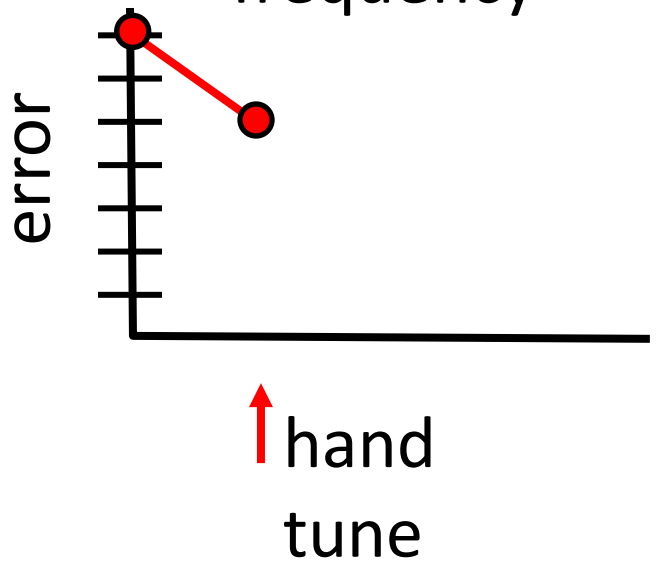
```
% setup fft, distance
dtildea = fft(dsta1);
x = rcsta1 - rcsta2;
alpha = 0.05;
c1 = 0.0001;
c2 = 0.9;
tau = 0.5;

% error and its gradient at the trial solution
gradE0 = zeros(Nx,1);
gradEg = zeros(Nx,1);
vx0 = vx;
vpos0 = interp1(fx,vx0,fpos,'linear','extrap');
vfull0 = [vpos0', fliplr( vpos0(2:Nf-1)' ) ]';
dv = 0.01;
dtildead0 = dtildea .* exp( i*x*w./vfull0 );
da0 = real(ifft( dtildead0 ));
e0 = c*dsta2 - da0;
E0 = e0'*e0;
for j=[1:Nx]
    vxp = vx0;
    vxp(j) = vxp(j)+dv;
    vposp = interp1(fx,vxp,fpos,'linear','extrap');
    vfullp = [vposp', fliplr( vposp(2:Nf-1)' ) ]';
    dtildeadp = dtildea .* exp( i*x*w./vfullp );
    dap = real(ifft( dtildeadp ));
    ep = c*dsta2 - dap;
    Ep = ep'*ep;
    gradE0(j) = (Ep-E0)/dv;
end

Niter=40;
for iter = [1:Niter]

    v = -gradE0 / sqrt(gradE0'*gradE0);

    % backstep
    for kk=[1:10]
        vxg = vx0+alpha*v;
        vposg = interp1(fx,vxg,fpos,'linear','extrap');
        vfullg = [vposg', fliplr( vposg(2:Nf-1)' ) ]';
        dtildeadg = dtildea .* exp( i*x*w./vfullg );
        dag = real(ifft( dtildeadg ));
        eg = c*dsta2 - dag;
        Eg = eg'*eg;
        for j=[1:Nx]
            vxp = vxg;
            vxp(j) = vxp(j)+dv;
            vposp = interp1(fx,vxp,fpos,'linear','extrap');
```

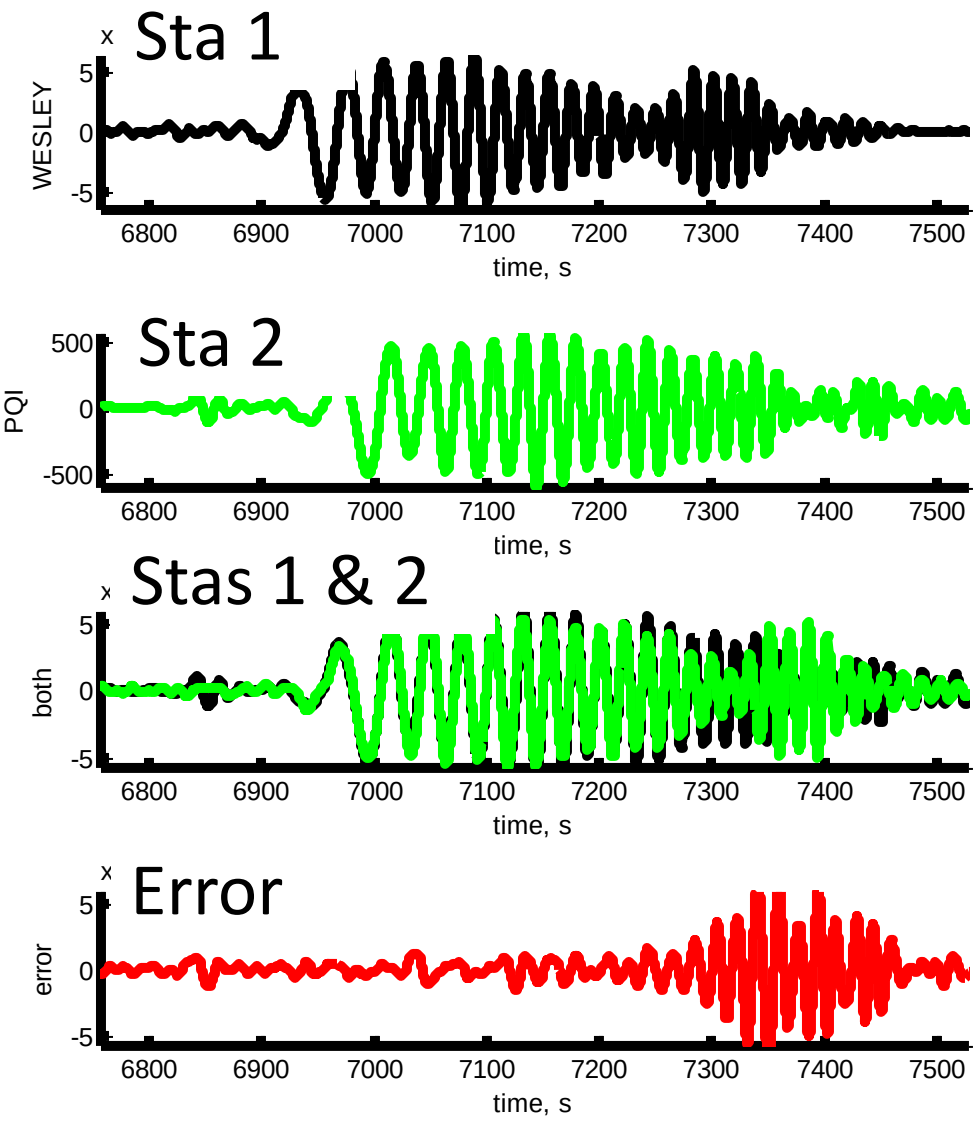
```
            vfullp = [vposp', fliplr( vposp(2:Nf-1)' ) ]';
            dtildeadp = dtildea .* exp( i*x*w./vfullp );
            dap = real(ifft( dtildeadp ));
            ep = c*dsta2 - dap;
            Ep = ep'*ep;
            gradEg(j) = (Ep-Eg)/dv;
        end
        if( (Eg<=(E0 + c1*alpha*v'*gradE0)) )
            break;
        end
        alpha = tau*alpha;
    end

    % change in solution
    Dvx = sqrt( (vxg-vx0)'*(vxg-vx0) );


    % update
    vx0=vxg;
    E0 = Eg;
    gradE0 = gradEg;

    if( Dvx < 1.0e-6 )
        break;
    end

    fprintf('Error for iteration %d is %e\n', iter, E0);


end
```

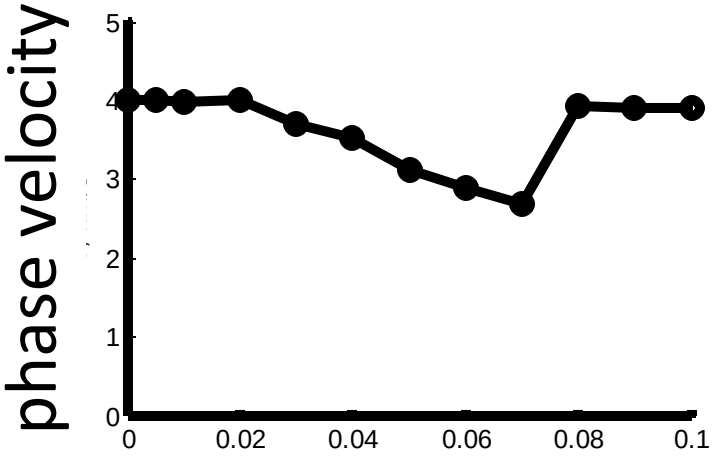

Sta 1
x
WESLEY
5
0
-5
6800
6900
7000
7100
7200
7300
7400
7500
time, s
Sta 2
PQI
500
0
-500
Stas 1 & 2
both
Error
error
phase velocity
frequency
error
start

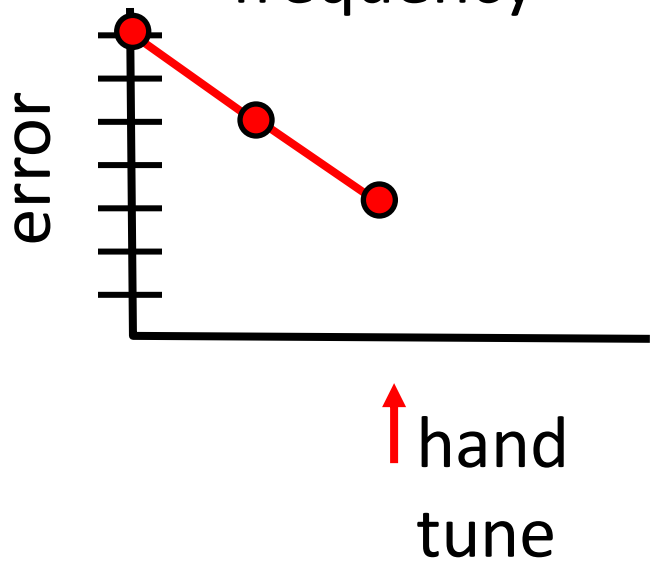

Sta 1
WESLEY
time, s
Sta 2
PQI
time, s
Stas 1 & 2
both
time, s
Error
error
time, s

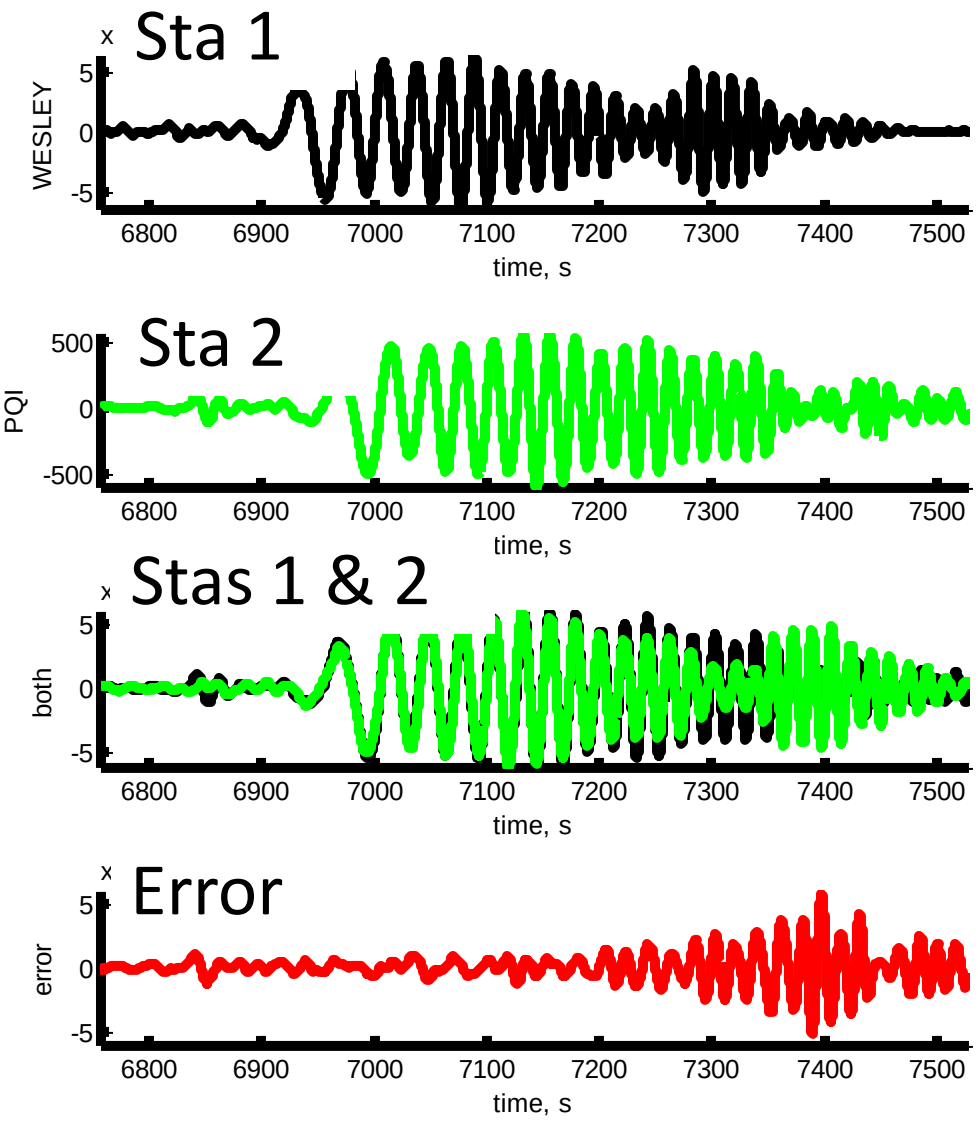

phase velocity
frequency

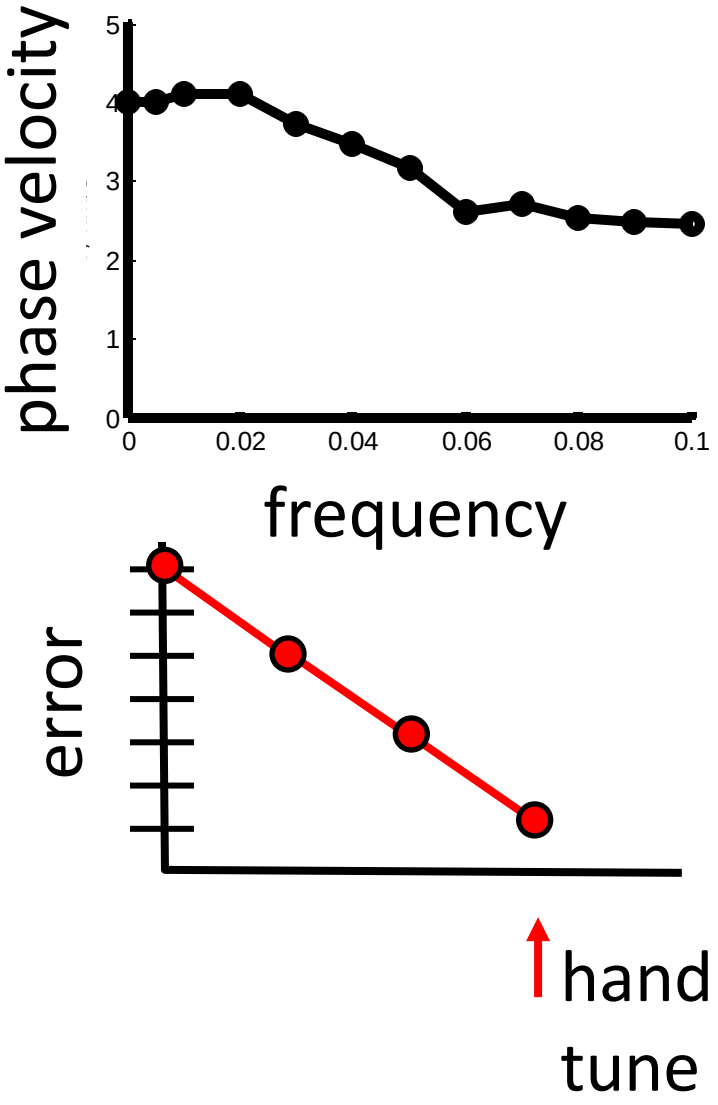

error
hand
tune

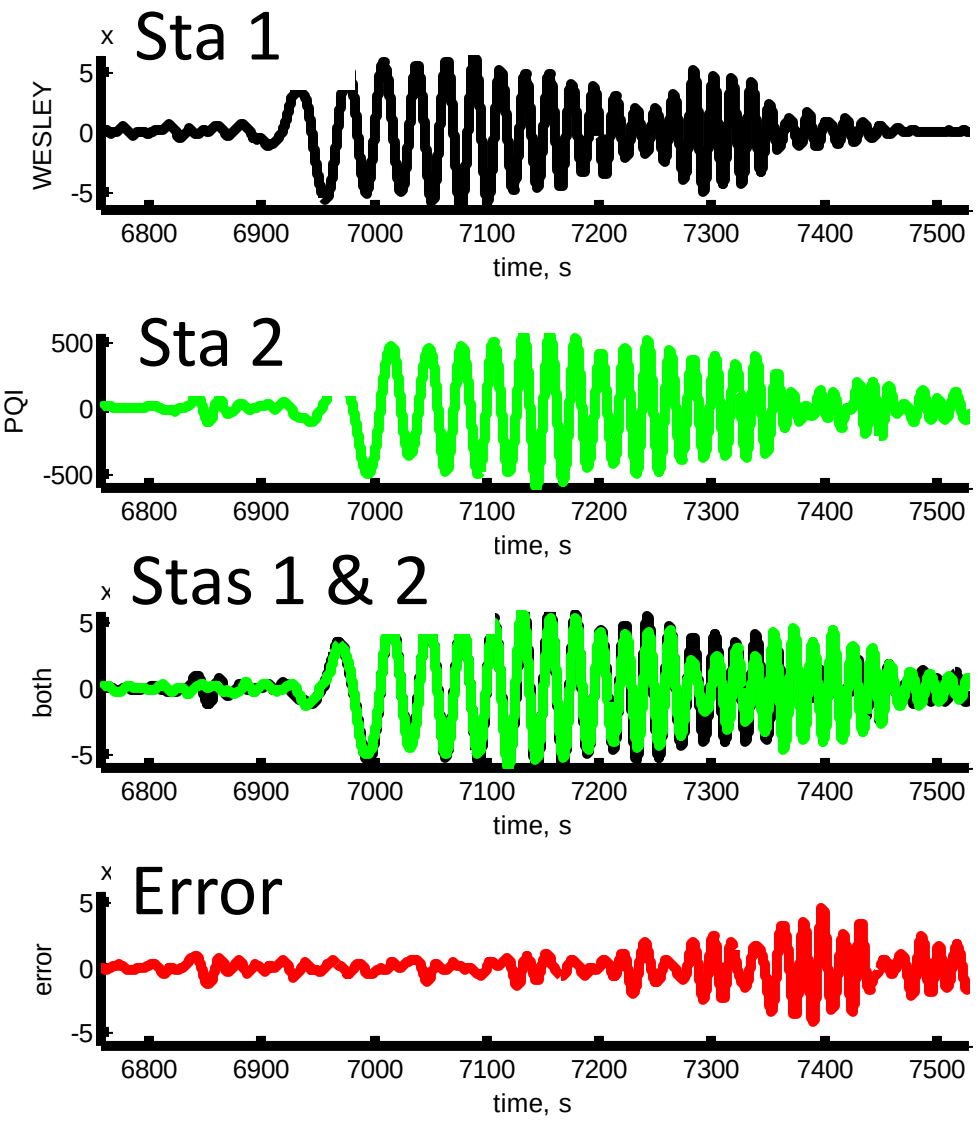

Sta 1
WESLEY
time, s
Sta 2
PQI
time, s
Stas 1 & 2
both
time, s
Error
error
time, s

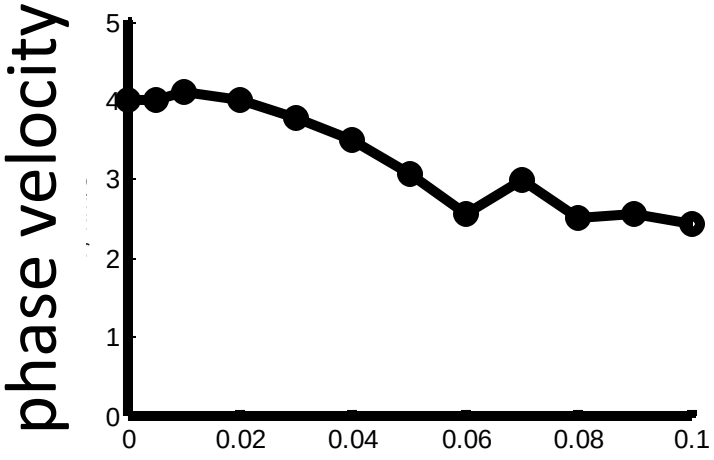

phase velocity
frequency

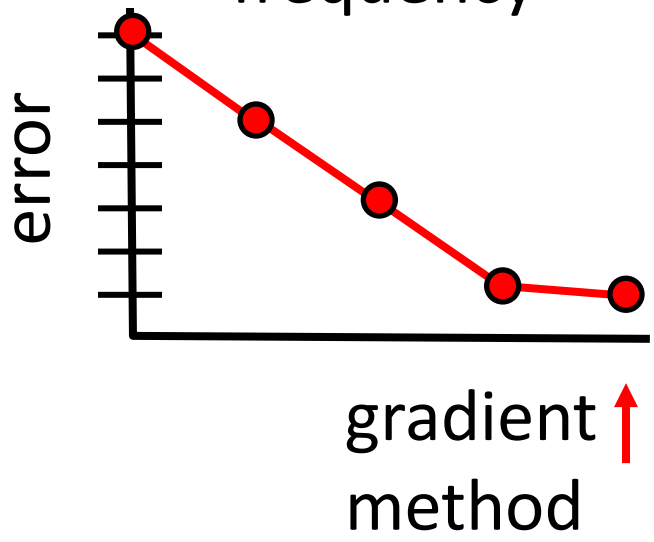

error
hand
tune


Sta 1
WESLEY
x 10^-5
5
0
-5
6800 6900 7000 7100 7200 7300 7400 7500
time, s
Sta 2
PQI
500
0
-500
6800 6900 7000 7100 7200 7300 7400 7500
time, s
Stas 1 & 2
both
x 10^-5
5
0
-5
6800 6900 7000 7100 7200 7300 7400 7500
time, s
Error
error
x 10^-5
5
0
-5
6800 6900 7000 7100 7200 7300 7400 7500
time, s


phase velocity
5
4
3
2
1
0
0 0.02 0.04 0.06 0.08 0.1
frequency
error
hand
tune


Sta 1
WESLEY
time, s
Sta 2
PQI
time, s
Stas 1 & 2
both
time, s
Error
error
time, s


phase velocity
frequency


error
gradient
method