```
% gda14_04
% This code parallels "Method Summary 3, Grid Search".

clear all;

% make synthetic data
% a function f(t) is composed of the sum of two
% sinusoids of unknown amplitude and frequency
% d(t) = A1 cos(2 pi f1 t) + B1 sin(2 pi f1 t)
%      = A2 cos(2 pi f2 t) + B2 sin(2 pi f2 t)

% true values
A1true = 0.9;
B1true = 0.5;
A2true = 0.2;
B2true = 0.3;
f1true = 7.1;
f2true = 11.4;

% time
N = 101;
Dt = 0.01;
t = Dt*[0:N-1]';

% true data and synthetic observed data
pi2 = 2*pi;
dtrue =       A1true*cos(pi2*f1true*t)+B1true*sin(pi2*f1true*t);
dtrue = dtrue+A2true*cos(pi2*f2true*t)+B2true*sin(pi2*f2true*t);
sigmad = 0.1;
sigmad2 = sigmad^2;
dobs = dtrue+random('Normal',0,sigmad, N, 1 );

% plot observed data
figure(1);
clf;
set(gca,'LineWidth',3);
hold on;
axis( [t(1), t(end), -2, 2] );
plot( t, dobs, 'ko', 'LineWidth', 2 );
plot( t, dobs, 'k-', 'LineWidth', 2 );
xlabel('time t (s)');
ylabel('d(t)');

% Step 1.  This problem is non-linear in the frequencies
% [f1, f2] but linear in the amplitudes [A1, B1, A2, B2].
% The strategy is to grid search over [f1, f2], computing
% [A1, B1, A2, B2] at each grid node via least squares.

% Step 2.  By visual inspection of the data, the frequencies
% of variation are about 10 Hz.  So we grid search in the
% (5-15) Hz band to be sure we enclose [f1, f2].  The
% data are a fairly slowly varying function of frequency,
% so a 201 by 201 grid should be sufficient.  Note that
% the problem is symmetrical in f1 and f2, so we could
% just search the f2>f1 part of the error surface E(f1,f2).
% However, coding this case seems more trouble than its
% worth.  We do have to be careful not to do the "f1 exactly
% equal to f2" case, because the least square data kernel
% is redundant in this case.  I avoid this problem by setting
```

```
% making the f1 and f2 grid points slightly different.
Nf1 = 201;
f1min=5.03;
f1max=15.0;
Df1=(f1max-f1min)/(Nf1-1);
f1list = f1min + Df1*[0:Nf1-1]';
Nf2 = 201;
f2min=5.05;
f2max=15.0;
Df2=(f1max-f1min)/(Nf2-1);
f2list = f2min + Df2*[0:Nf2-1]';

% Step 3: The Grid Searh
E = zeros( Nf1, Nf2 );
M = 6;
for if1 = [1:Nf1] % loop over values of f1
for if2 = [1:Nf2] % loop over values of f2
    f1 = f1list(if1); % set [f1, f2]
    f2 = f2list(if2);
    % create data kernel for m=[A1, B1, A2, B2] for fixed [f1, f2]
    G = [ cos(pi2*f1*t), sin(pi2*f1*t), cos(pi2*f2*t), sin(pi2*f2*t) ];
    mest = (G'*G)\(G'*dobs); % least squares solution
    dpre = G*mest; % predicted data
    e = dobs - dpre; % prediction error
    E(if1,if2) = e'*e; % tabulate overall error
end
end

% refine minimum using quadratic approximation.  Note that I am
% using the prior veriance of the data in the calculation of covariance
[ f1est, f2est, E0, covf1f2, status ] = gda_Esurface( f2list, f1list, E, sigmad2 );

% recompute m=[A1, B1, A2, B2] for the refined [f1, f2]
G = [ cos(pi2*f1est*t), sin(pi2*f1est*t), cos(pi2*f2est*t), sin(pi2*f2est*t) ];
mest = (G'*G)\(G'*dobs);
dpre = G*mest;
e = dobs - dpre;
Efinal = e'*e;

sigma2dest = Efinal/(N-M); % posterior variance in the data
% recompute the covariance of frequencies based on the posterior variance
[ f1est, f2est, E0, covf1f2, status ] = gda_Esurface( f1list, f2list, E, sigma2dest );
sigmaf1 = sqrt( covf1f2(1,1) );
sigmaf2 = sqrt( covf1f2(2,2) );
% write out the frequencies and their 95% confidence intervals
fprintf('f1: %.3f +/- %.3f (95%%)\n', f1est, 2*sigmaf1 );
```

```
f1: 7.065 +/- 0.015 (95%)
```

```
fprintf('f2: %.3f +/- %.3f (95%%)\n', f2est, 2*sigmaf2 );
```

```
f2: 11.424 +/- 0.044 (95%)
```

```
% Use the standard least squares estimate of the covariance of the
% amplitudes (using the posterior variance of the data)
covm = sigma2dest * inv(G'*G);
A1est = mest(1); B1est=mest(2); A2est = mest(3); B2est=mest(4);
sigmaA1 = sqrt( covm(1,1) );
sigmaB1 = sqrt( covm(2,2) );
```

```
sigmaA2 = sqrt( covm(3,3) );
sigmaB2 = sqrt( covm(4,4) );
% write out the amplitudes and their 95% confidence intervals
fprintf('A1: %.3f +/- %.3f (95%%)\n', A1est, 2*sigmaA1 );
```

```
A1: 0.945 +/- 0.028 (95%)
```

```
fprintf('B1: %.3f +/- %.3f (95%%)\n', B1est, 2*sigmaB1 );
```

```
B1: 0.473 +/- 0.029 (95%)
```

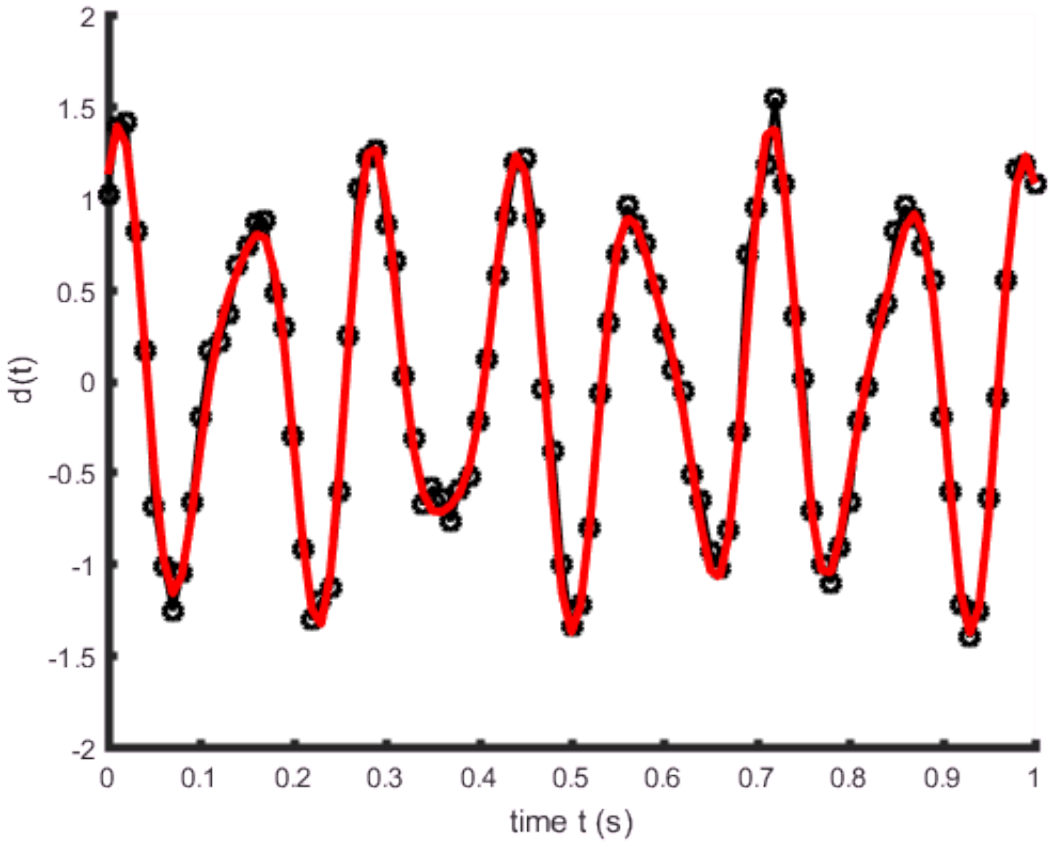
```
fprintf('A2: %.3f +/- %.3f (95%%)\n', A2est, 2*sigmaA2 );
```

```
A2: 0.181 +/- 0.028 (95%)
```

```
fprintf('B2: %.3f +/- %.3f (95%%)\n', B2est, 2*sigmaB2 );
```

```
B2: 0.299 +/- 0.028 (95%)
```

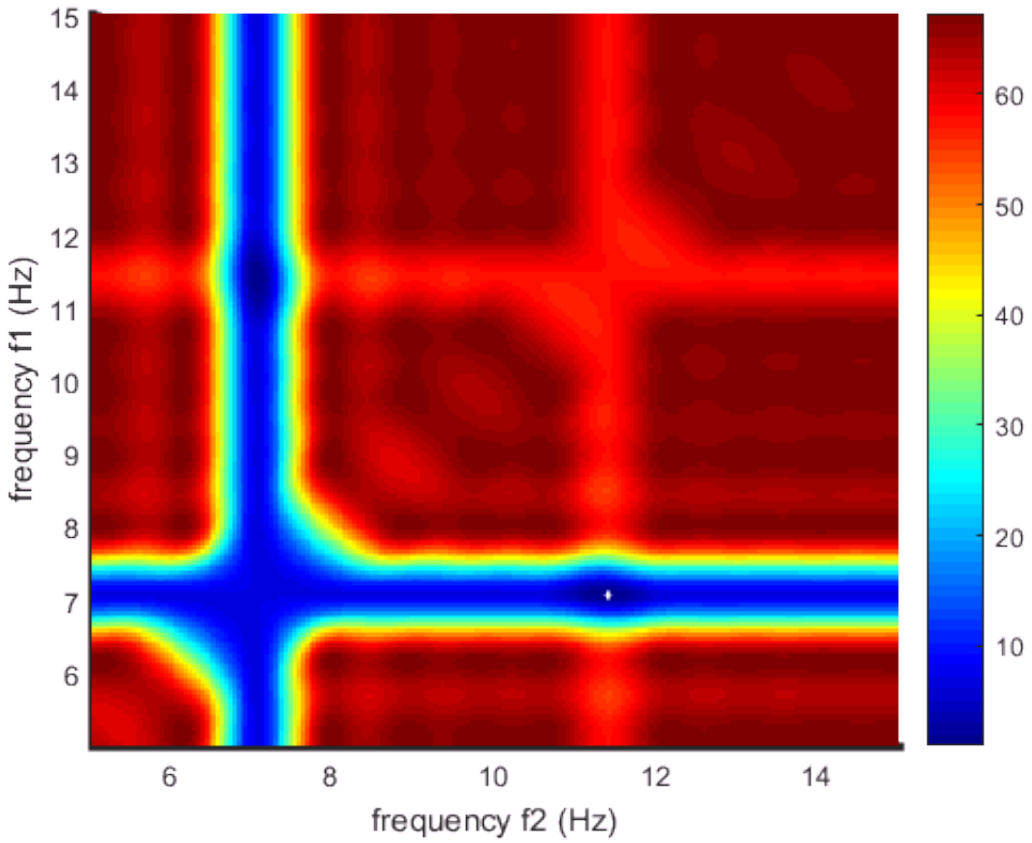
```
% plot the predicted data
plot( t, dpre, 'r-', 'LineWidth', 3 );
```




```
% Figure 1: Observed data d(t) and its prediction m(t) as a function
% of time t

% plot of error surface and estimated frequencies
figure(2);
```

```
clf;
set(gca,'LineWidth',3);
hold on;
axis xy;
axis( [f1min, f1max, f2min, f2max] );
colormap('jet');
colorbar();
% plot the error surface.  Note the horizontal stripes
% on the image parallel to an f1 or f2 of 7 Hz. This
% frequency has the dominant amplitude, so when one
% of the frequenices is about 7 Hz, the prediction will
% fit the data to a reasonable degree. As expected, the
% error surface is symmtrical about the f1=f2 line.  We
% anticipated this behavior (see above).
imagesc( [f2min, f2max], [f1min, f1max], E );
xlabel('frequency f2 (Hz)');
ylabel('frequency f1 (Hz)');
% plot estimated frequencies and their 95% confidence intervals
plot( [f2est-sigmaf2, f2est+sigmaf2]', [f1est,f1est]', 'w-', 'LineWidth', 3 );
plot( [f2est, f2est]', [f1est-sigmaf1,f1est+sigmaf1]', 'w-', 'LineWidth', 3 );
```




```
% Figure 2. Error surface (colors) showing point of minimum error
% (dot) and error bars (white lines). You will need to zoom the image
% so see the error bars.  Note that the f2 bar - the one
% parallel to the trough in the error surface - is the larger.
% Note that the error surface is has mirror symmetry around the
% line f1=f2.  That's because of a non-uniqueness in the problem:
% the two frequencies (and corresponding amplitudes) can be
% interchanged without changing the error.
```