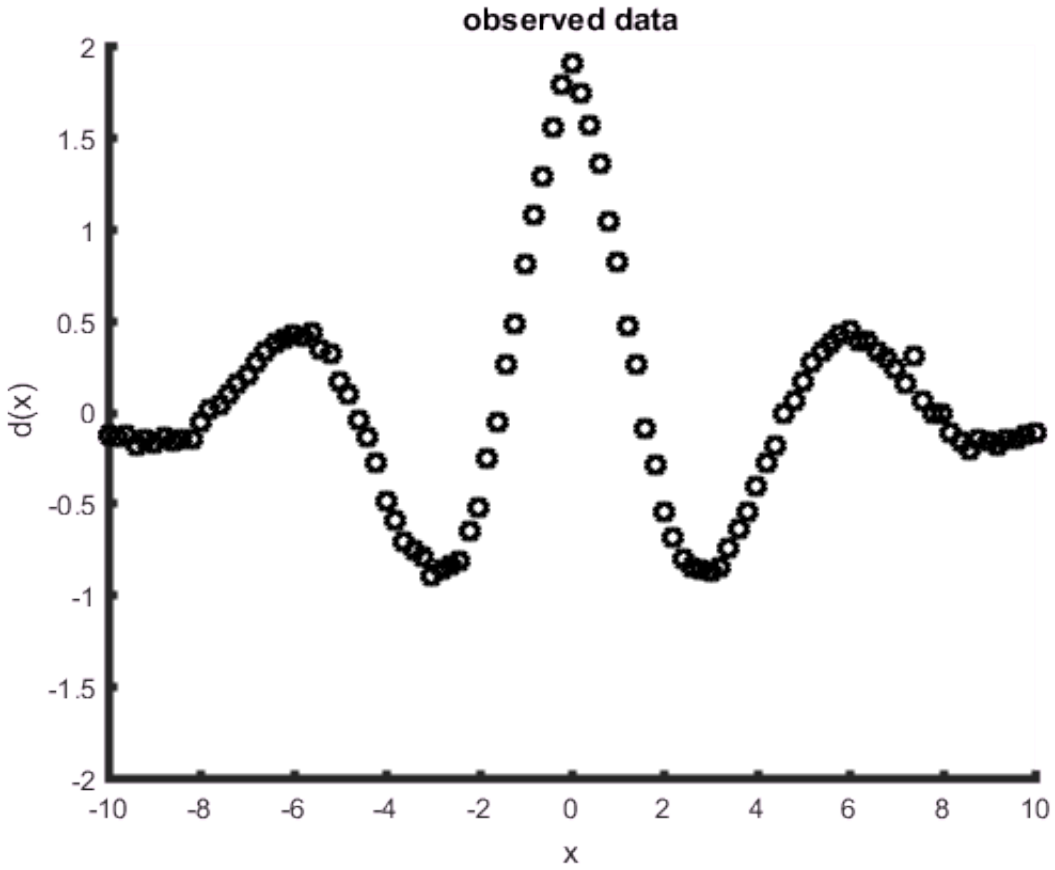
```
% gda14_02
% This code parallels "Method Summary 1, Least Squares"
clear all;

% global necessary to use biconjugate gradient solver
global F;

% Step 1: State the problem in words.
% This is a data smoothing problem where the model
% parameters are a discretized version of a function
% m(x) (at equal increments Dx) and the data are noisy
% measurements of the same model function.

% set up the auxiliary variable, x
Nx=101;
xmin = -10;
xmax = 10;
xL = (xmax-xmin);
Dx = xL/(Nx-1);
Dx2i = 1/(Dx^2);
x = xmin + Dx*[0:Nx-1]';

% Step 2: Organize the problem in standard form
% number of model parameters = number of data = Nx
% and the data kernel is the identity matrix, G=I
N=Nx;
M=N;
G = speye(N,M);

% this is a synthetic test, so there is no real data
% the true model is a "ringy" function peaked at x=0
% the observed data is the true model plus random noise
Atrue = 1.9; % amplitude
Ltrue=6.21; % wavelength
b=4; % decay parameter
mtrue=Atrue*exp(-2*pi*abs(x)/(b*Ltrue)).*cos(2*pi*x/Ltrue);
dtrue = mtrue;
sigmad = 0.03; % noise level of the observations
sigma2d = sigmad^2;
dobs = dtrue+random('Normal',0,sigmad,N,1);
No=floor(7*N/8); % create an outlier in the data
dobs(No) = 0.3;

% Step 3: Plot the data
% When you examine the plot, you should take notice of:
%   the overall shape of the curve
%   the noise level
%   the existence of an outlier toward the right hand side
figure(1);
clf;
set(gca,'LineWidth',3);
hold on;
axis( [xmin, xmax, -2, 2] );
plot( x, dobs, 'ko', 'LineWidth', 2 );
xlabel('x');
ylabel('d(x)');
title('observed data');
```




```
% Figure 1. The observed data, which sample a smooth curve d(x)

% I'm going to remove the outlier "manually"
% by interpolating nearby points
dobs(No) = 0.5*(dobs(No-1)+dobs(No+1));

% Step 4: Establish the accuracy of the data
% In this case, we know the sigmad, because it
% was specified when the synthetic data were
% created. I'm going to use this "correct" value

% Step 5: State the prior information in words.
% We assume that the function is smooth, meaning
% that its second derivative is small

% Step 6: Organize the prior information in standard
% form, H is a first differnce operator, h=0.  There
% are two few rows in H than there are model parameters
K=M-2;
z = Dx2i*ones(M,1);
H = spdiags([z, -2*z, z], [0, 1, 2], K, M);
h = zeros(K,1);

% Step 7: Establish the accuracy of the a priori information
% Suppose that I know that the function is roughly sinusoidal
% with an amplitude of about A and a wavelength of about L.
% Then if the function were a cosine wave m = A cos(2 pi x / L)
% its second derivative would be d2m/dx2 = -A (2 pi/ L)^2
% cos(2 pi x / L). The variance of the second derivative
% would therefore be about 0.5 A^2 (2 pi/ L)^4 (since the
% average value of cos^2 is 0.5).  So I'm going to use this
```

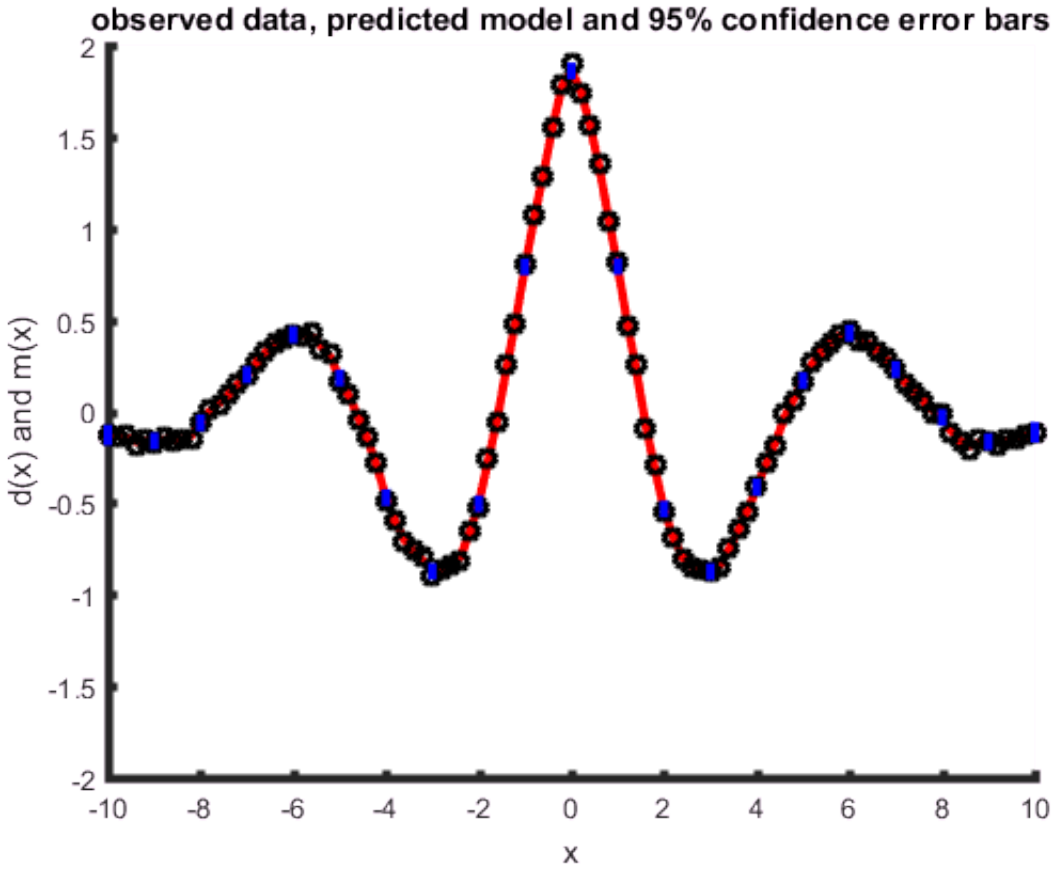
```
% amount as an estimate of the variance of the prior information;
% that is, the second derivative is zero +/- sigmah.
A = 2.0;
L = 6.0;
sigma2h = 0.5 * (A^2) * ((2*pi/L)^4);
sigmah = sqrt(sigma2h);

% Step 8: Estimate model parameter and their covariance
% set up the matrix F and vector f
F=spalloc(N+K,M,3*(N+K));
f=zeros(N+K,1);
k=1; % this is a trick to count rows of F
for i=[1:N] % the G, dobs part
    F(k,i) = 1/sigmad;
    f(k) = dobs(i)/sigmad;
    k = k+1;
end
for i=[1:K] % The H, h=0 part
    F(k,i) = Dx2i/sigmah;
    F(k,i+1) = -2*Dx2i/sigmah;
    F(k,i+2) = Dx2i/sigmah;
    f(k)=0;
    k = k+1;
end

% solve Fm=f by biconjugate gradients
mest = bicg( @weightedleastsquaresfcn, F'*f, 1e-5, 4*(N+K) );
```

```
bicg converged at iteration 10 to a solution with relative residual 4.2e-06.
```

```
% plot the solution
% note that it fits the observations pretty well
figure(2);
clf;
set(gca,'LineWidth',3);
hold on;
axis( [xmin, xmax, -2, 2] );
plot( x, mest, 'r-', 'LineWidth', 3 );
plot( x, dobs, 'ko', 'LineWidth', 2 );
xlabel('x');
ylabel('d(x) and m(x)');
title('observed data, predicted model and 95% confidence error bars');

% Step 9: (see a little further, below)

% Step 10: Confidence interval calulation
% The covariance matrix is [cov m] = inv(F'*F)
% However, we do not necessarily need to compute the complete
% matrix.  The code here solves for the k-th column, say ck,
% (or row, since it is symmetric).  The idea is to solve the equation
% (F'F) ck = s, where s is a vector that is all zeros except
% a 1 in row k.  Then ck is the k-th column of inv(F'*F)
varlist=[1:floor(N/20):N];  % list of values of x at which
                            % to calculate error bars
for k = varlist
    % solve (F'F) ck = s
    s = zeros(M,1);
    s(k)=1;
    ck = bicg( @weightedleastsquaresfcn, s, 1e-5, 4*M );
```

```
    % variance of estimated model parameters is k-th element of ck
    sigmamest = sqrt(ck(k));

    % 95% confidence interval
    mlow = mest(k)-2*sigmamest;
    mhigh = mest(k)+2*sigmamest;

    % plot error bars
    plot( [x(k), x(k)], [mlow, mhigh], 'b-', 'LineWidth', 3 );

end
```

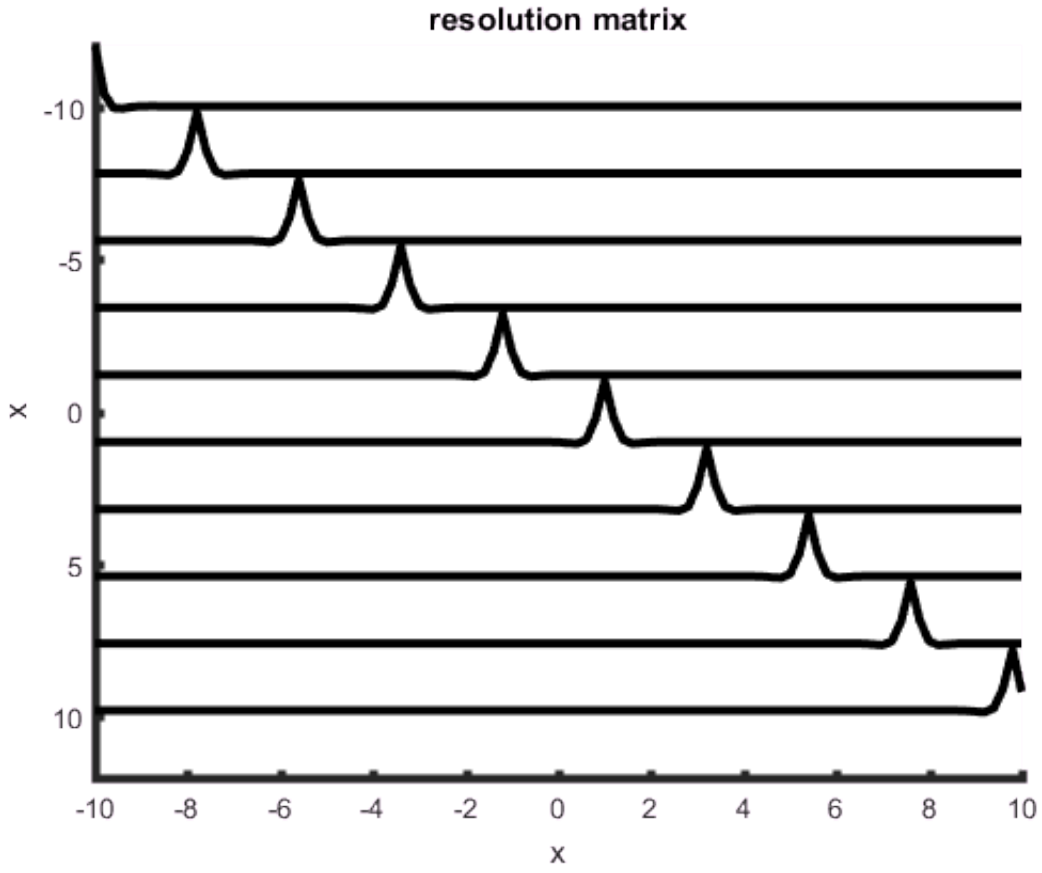
```
bicg converged at iteration 12 to a solution with relative residual 6.3e-06.
bicg converged at iteration 12 to a solution with relative residual 7.8e-06.
bicg converged at iteration 12 to a solution with relative residual 8.4e-06.
bicg converged at iteration 13 to a solution with relative residual 3.1e-06.
bicg converged at iteration 12 to a solution with relative residual 1e-05.
bicg converged at iteration 12 to a solution with relative residual 1e-05.
bicg converged at iteration 12 to a solution with relative residual 1e-05.
bicg converged at iteration 12 to a solution with relative residual 1e-05.
bicg converged at iteration 12 to a solution with relative residual 1e-05.
bicg converged at iteration 12 to a solution with relative residual 1e-05.
bicg converged at iteration 12 to a solution with relative residual 1e-05.
bicg converged at iteration 12 to a solution with relative residual 1e-05.
bicg converged at iteration 12 to a solution with relative residual 1e-05.
bicg converged at iteration 12 to a solution with relative residual 1e-05.
bicg converged at iteration 12 to a solution with relative residual 1e-05.
bicg converged at iteration 12 to a solution with relative residual 1e-05.
bicg converged at iteration 12 to a solution with relative residual 1e-05.
bicg converged at iteration 12 to a solution with relative residual 1e-05.
bicg converged at iteration 13 to a solution with relative residual 3.1e-06.
bicg converged at iteration 12 to a solution with relative residual 8.4e-06.
bicg converged at iteration 12 to a solution with relative residual 7.8e-06.
bicg converged at iteration 12 to a solution with relative residual 6.3e-06.
```




```
% Figure 2. Observed data (circles), predicted model (red) and 95% conficence error bars (blue

% Note that the error bars are a small fraction of the
% overall range of the function, meaning that we haven't
% smoothed the function completely away!

% Step 9: Examine the model resolution
% The resolution matrix is RG = GMG * G
% with GMG = inv(F'F) G' inv(covd)
% However, as with covariance, one does not need to
% compute every row.  This code just does a few rows.
% Note that the plot shows that the resolution is
% peaked along the main diagonal, but that it has a
% finite width of about 1 (meaning that there is some
% smoothing (but that's what we wanted) and a little
% tiny bit of negative sidelobes (which is not so
% good, but unavoidable with 2nd derivative smoothing).
reslist=[1:floor(N/9):N];  % do these rows
Nres=length(reslist);
figure(3);
clf;
axis ij;
set(gca,'LineWidth', 3);
hold on;
axis( [xmin, xmax, xmin-(xmax-xmin)/Nres, xmax+(xmax-xmin)/Nres] );
xlabel('x');
ylabel('x');
title('resolution matrix');
vard = sigma2d * ones(N,1);  % variance of data
% (the code can handle the case where it varies)
for k = reslist
```

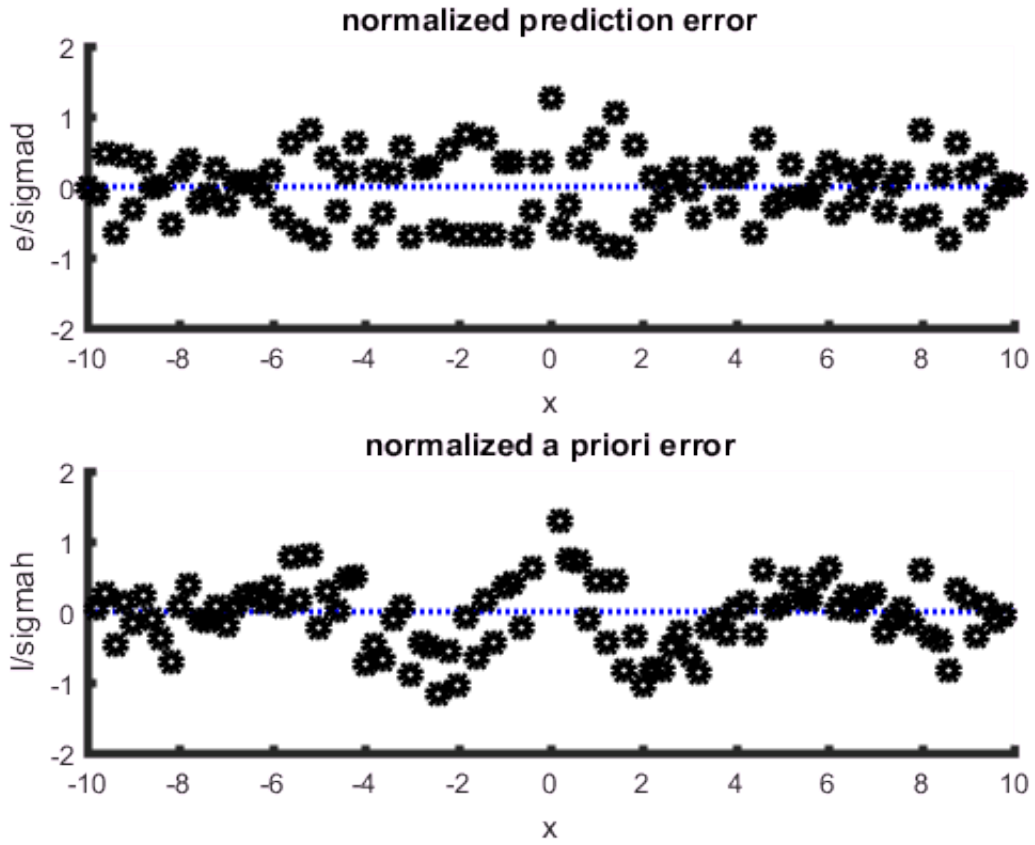
```
    % compute k-th row of the resolution matrix
    % first: compute column of inverse of F'F
    % note F'F symmetric, so column is also row
    s = zeros(M,1);
    s(k)=1;
    ck = bicg( @weightedleastsquaresfcn, s, 1e-5, 4*M );
    % second: row of generalized inverse GMG(k,:)=gi
    gi = ((ck')*(G'))./(vard');
    % third: row of the resolution matrix
    r_row = gi*G;
    sc = -((xmax-xmin)/Nres)/max(r_row);
    plot( x, sc*r_row'+x(k), 'k-', 'LineWidth', 3 );
end
```

```
bicg converged at iteration 12 to a solution with relative residual 6.3e-06.
bicg converged at iteration 13 to a solution with relative residual 3.4e-06.
bicg converged at iteration 12 to a solution with relative residual 1e-05.
bicg converged at iteration 12 to a solution with relative residual 1e-05.
bicg converged at iteration 12 to a solution with relative residual 1e-05.
bicg converged at iteration 12 to a solution with relative residual 1e-05.
bicg converged at iteration 12 to a solution with relative residual 1e-05.
bicg converged at iteration 12 to a solution with relative residual 1e-05.
bicg converged at iteration 12 to a solution with relative residual 8.6e-06.
bicg converged at iteration 12 to a solution with relative residual 6.4e-06.
```



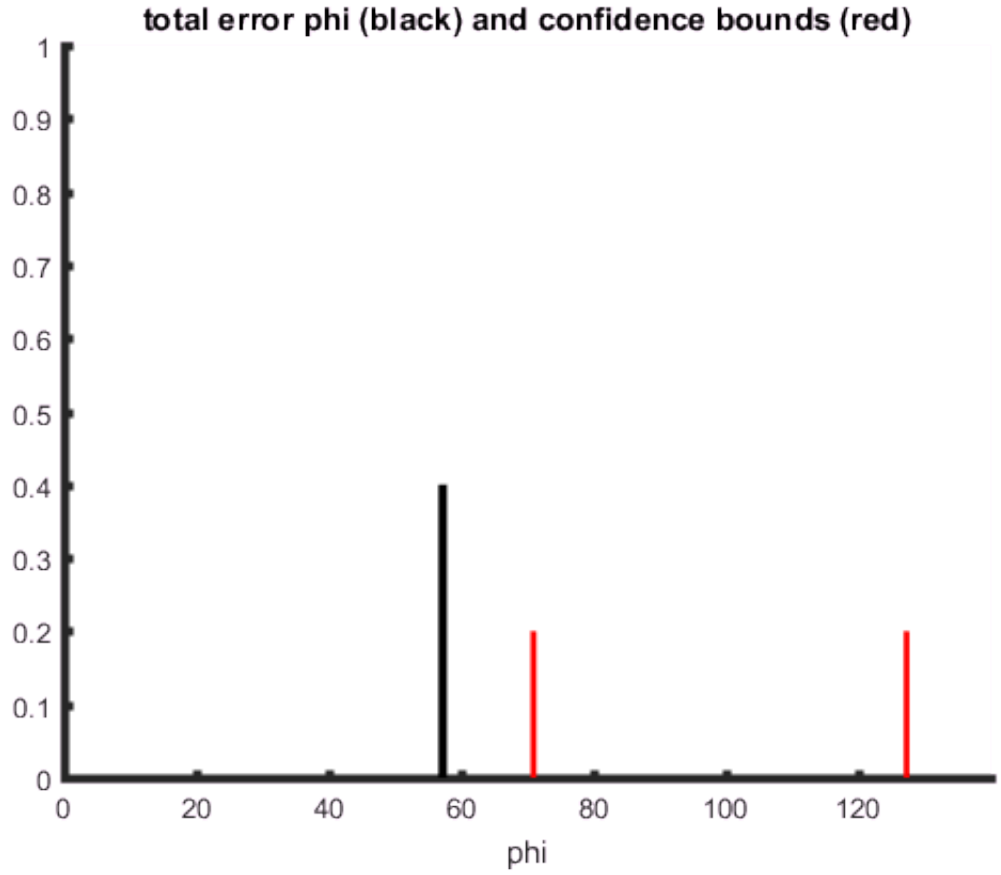

```
% Figure 3: Selected rowas of the model reolution matrix.  Note that
% it is peaked near the main diagonal (good), but also has a small
% negative sidelobes (bad)

% Step 11: Examine the individual errors
```

```
% Note that the prediction error is of
% uniform size of about unity, indicating
% that our a priori data variance is about
% right.  It is also of similar size across
% the plot, indicating that our assumption
% that the error is uniform is about right,
% too. One the other hand, the error
% in a priori information is a little less
% than unity and not uniform across the
% plot, indicating that our assumptions on
% the size and uniformity of sigmah is not
% quite right.

% Plot prediction error
dpre = G*mest;
e = dobs-dpre;
figure(4);
clf;
subplot(2,1,1);
set(gca,'LineWidth', 3);
hold on;
axis( [xmin, xmax, -2, 2] );
plot( [xmin, xmax]', [0, 0]', 'b:', 'LineWidth', 2);
plot( x, e/sigmad, 'ko', 'LineWidth', 3 );
xlabel('x');
ylabel('e/sigmad');
title('normalized prediction error');

% plot a priori error
hpre = H*mest;
l = h-hpre;
subplot(2,1,2);
set(gca,'LineWidth', 3);
hold on;
axis( [xmin, xmax, -2, 2] );
plot( [xmin, xmax]', [0, 0]', 'b:', 'LineWidth', 2);
plot( x(2:N-1), l/sigmah, 'ko', 'LineWidth', 3 );
xlabel('x');
ylabel('l/sigmah');
title('normalized a priori error');
```




```
% Figure 4. (Top) The normalized prediction error randomly scatters
% around zero with an amplitude of +/- one (good).  The normalized
% error in prior information scatters is less random (bas) expecially
% near x=0.

% Step 12: Examine the total error phiest
% in order to address the null hypothesis
% that departure from the expected value of
% nu is due to random error.
% In the plot, total error phiest is usually
% close to the minimum bound (sometimes on
% one side, sometimes on the other), indicating
% that one of the two variances, sigma2d or
% sigma2h is overestimated. (Our estimate of
% sigma2h is probably too large. Multiplying
% it by 0.75 produces a better result).
fpre = F*mest;
phiest = (f-fpre)'*(f-fpre);
nu = N+K-M;
p1 = nu - 2*sqrt(2*nu);
p2 = nu + 2*sqrt(2*nu);

figure(5);
clf;
set(gca,'LineWidth',3);
hold on;
pmax = max([phiest,p2]);
axis( [0, 1.1*pmax, 0, 1] );
plot( [p1, p1]', [0, 0.2]', 'r-', 'LineWidth', 2);
plot( [p2, p2]', [0, 0.2]', 'r-', 'LineWidth', 2);
plot( [phiest, phiest]', [0, 0.4]', 'k-', 'LineWidth', 3);
```

```
xlabel('phi');
title('total error phi (black) and confidence bounds (red)');
```




```
% Figure 5.  The observed phi (black bar), together with the 95%
% confidence interval (red bars) in which phi is predicted to lie
% presuming that our assumptions about the size of the measurement
% error and the error in prior information are correct.

% Step 13: Two different models, A and B?
% We have only one model, so this step is not relevant
```