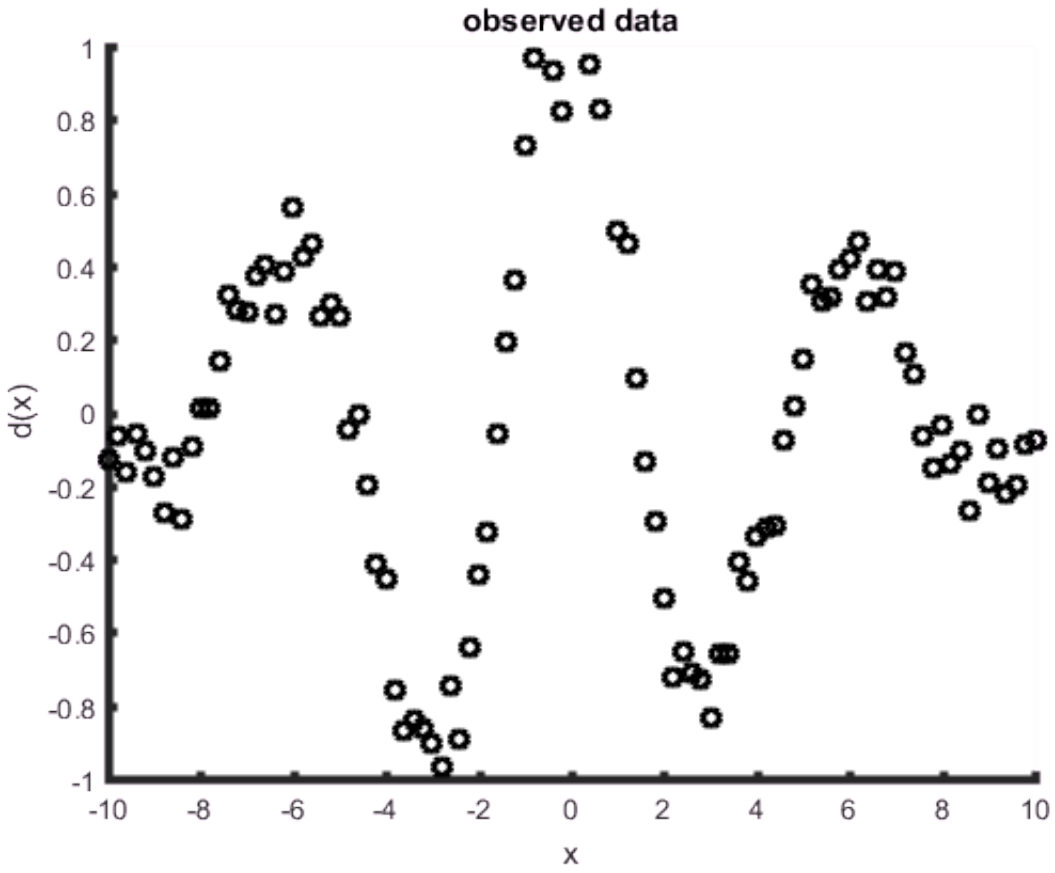
```
% gda14_03
% This code parallels "Method Summary 2, Nonlinear Least Squares"
clear all;

% global necessary to use biconjugate gradient solver
global F;

% Step 1: State the problem in words.
% This is a simple modification of the data smoothing
% solve in the example for Methods Summary 1, Least
% Squares.  As before, the the model parameters are
% a discretized version of a function m(x) (at equal
% increments Dx).  But now the the data are nonlinear
% functions of each model parameter d(i) = erf( m(i) ).
% The effor function erf(z) is sigmoidal in shape.  Near
% the origin, erf(z)~z, but for large |z| it asymtopes
% to +/- 1.  It thus squeezes the larger model parameters
% into a small range.  The derivative is d erf(z) / dz
% = (2/sqt(pi)) exp(-z^2)

% 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;

% 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 = erf(mtrue);
sigmad = 0.1; % noise level of the observations
sigma2d = sigmad^2;
dobs = dtrue+random('Normal',0,sigmad,N,1);

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

% Plot the data
% When you examine the plot, you should take notice of:
%   the overall shape of the curve
%   the noise level
%   the shape of the central peak, which is rather flat
%     (which is due to the nonlinearity; information about
```

```
%     the height of the central peak is being lost, and
%     that will limit the success of the inversion).
figure(1);
clf;
set(gca,'LineWidth',3);
hold on;
axis( [xmin, xmax, -1, 1] );
plot( x, dobs, 'ko', 'LineWidth', 2 );
xlabel('x');
ylabel('d(x)');
title('observed data');
```




```
% Figure 1: The observed data samples a smooth curve d(x)

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

% 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);

% 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 3: Decide up a reasonable trial solution
% Let's try zeros
mk = zeros(M,1);

% Step 4: Linearized the data equation

% iterations
Niter = 100; % Maximum number of iterations
for iter = [1:Niter]

% perturbations in data and prior information
Dd = dobs - erf(mk);
Dh = h - H*mk;

% The data kernel G is diagonal
dddm = (2/sqrt(pi)) * exp(-mk.^2);
G = spdiags(dddm, 0, N, M);

% Step 5: Form the combined equation
% 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) = dddm(i)/sigmad;
    f(k) = Dd(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) = Dh(i)/sigmah;
    k = k+1;
end

% Step 6: Iteratively improve the solution
% solve F Dm = f by biconjugate gradients
Dm = bicg( @weightedleastsquaresfcn, F'*f, 1e-5, 4*(N+K) );
% update solution
mk = mk + Dm;

% Step 7: Stop iterating
% terminate for very small Dm
Sm = (Dm'*Dm)/M;
Smlimit = 1E-6;
if( Sm < Smlimit )
    break;
end
```

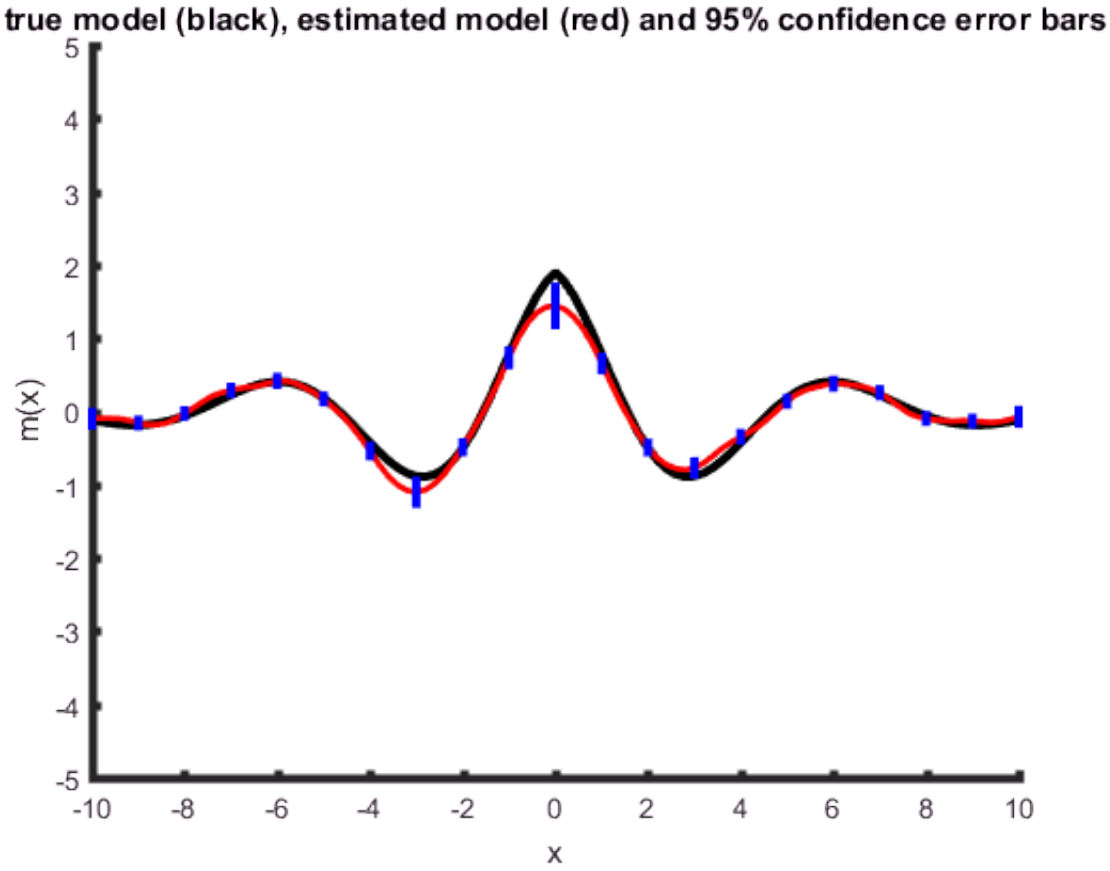
```
end % next iteration
```

```
bicg converged at iteration 31 to a solution with relative residual 8.8e-06.
bicg converged at iteration 59 to a solution with relative residual 9.8e-06.
bicg converged at iteration 75 to a solution with relative residual 5.8e-06.
bicg converged at iteration 76 to a solution with relative residual 6.4e-06.
bicg converged at iteration 76 to a solution with relative residual 5.8e-06.
```

```
mest = mk;

% Step 8: Examine the results

% plot the solution
% note that the solution is not able to fit
% the true central peak very well
figure(2);
clf;
set(gca,'LineWidth',3);
hold on;
axis( [xmin, xmax, -5, 5] );
plot( x, mtrue, 'k-', 'LineWidth', 3 );
plot( x, mest, 'r-', 'LineWidth', 2 );
xlabel('x');
ylabel('m(x)');
title('true model (black), estimated model (red) and 95% confidence error bars');

% 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 35 to a solution with relative residual 9.4e-06.
bicg converged at iteration 35 to a solution with relative residual 9.8e-06.
bicg converged at iteration 37 to a solution with relative residual 9.4e-06.
bicg converged at iteration 42 to a solution with relative residual 9.5e-06.
bicg converged at iteration 51 to a solution with relative residual 9.4e-06.
```

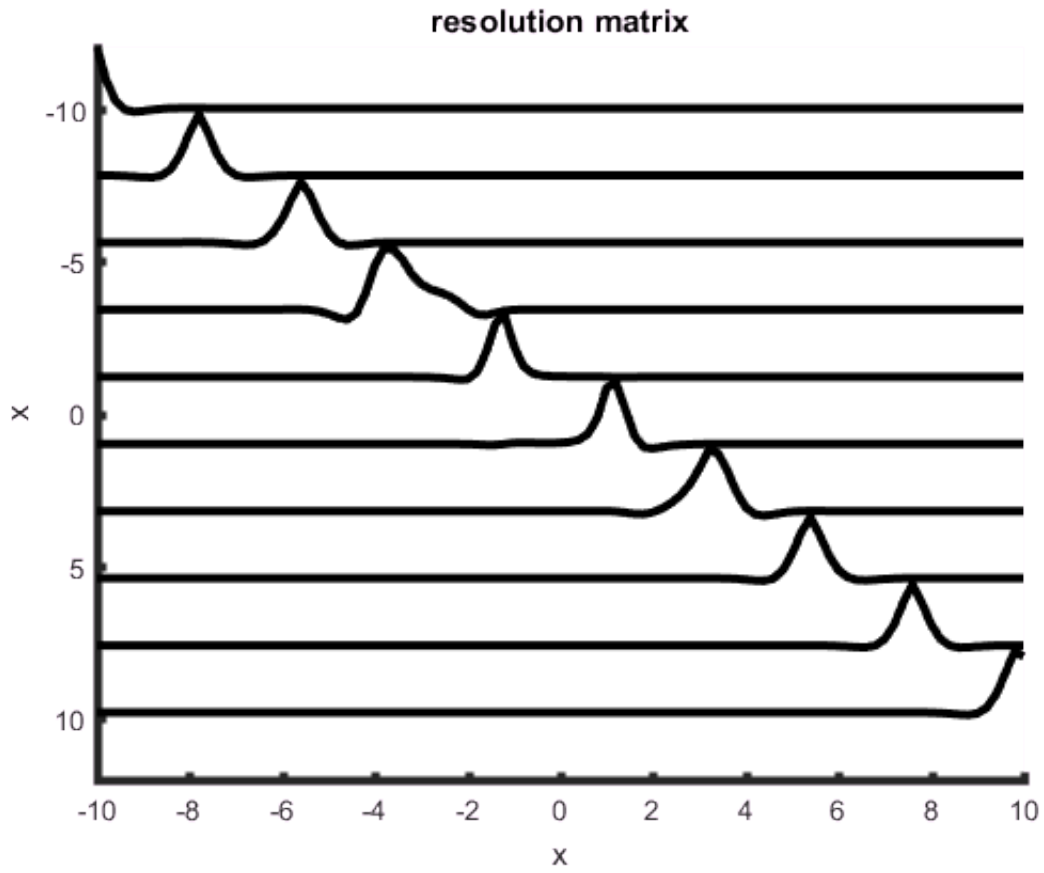
```
bicg converged at iteration 61 to a solution with relative residual 9.4e-06.
bicg converged at iteration 75 to a solution with relative residual 8.4e-06.
bicg converged at iteration 75 to a solution with relative residual 5.7e-06.
bicg converged at iteration 75 to a solution with relative residual 8.5e-06.
bicg converged at iteration 76 to a solution with relative residual 9.8e-06.
bicg converged at iteration 75 to a solution with relative residual 8.1e-06.
bicg converged at iteration 77 to a solution with relative residual 6.3e-06.
bicg converged at iteration 76 to a solution with relative residual 6.2e-06.
bicg converged at iteration 70 to a solution with relative residual 8.8e-06.
bicg converged at iteration 55 to a solution with relative residual 7.8e-06.
bicg converged at iteration 47 to a solution with relative residual 9.4e-06.
bicg converged at iteration 44 to a solution with relative residual 7.9e-06.
bicg converged at iteration 39 to a solution with relative residual 6.9e-06.
bicg converged at iteration 37 to a solution with relative residual 8.6e-06.
bicg converged at iteration 35 to a solution with relative residual 8.9e-06.
bicg converged at iteration 36 to a solution with relative residual 7.3e-06.
```




```
% Figure 2. The true function d(x) (black) and the reconstructed
% function m(x) (red), together with 95% confidence
% error bars (blue). Note that the error bars scale with the size of
% the solution.  That's due to the erf() nonlinearity,
% that makes difficult distinguishing a m, say, of 2 from a m say of 3.

% 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
% mostly peaked along the main diagonal.
```

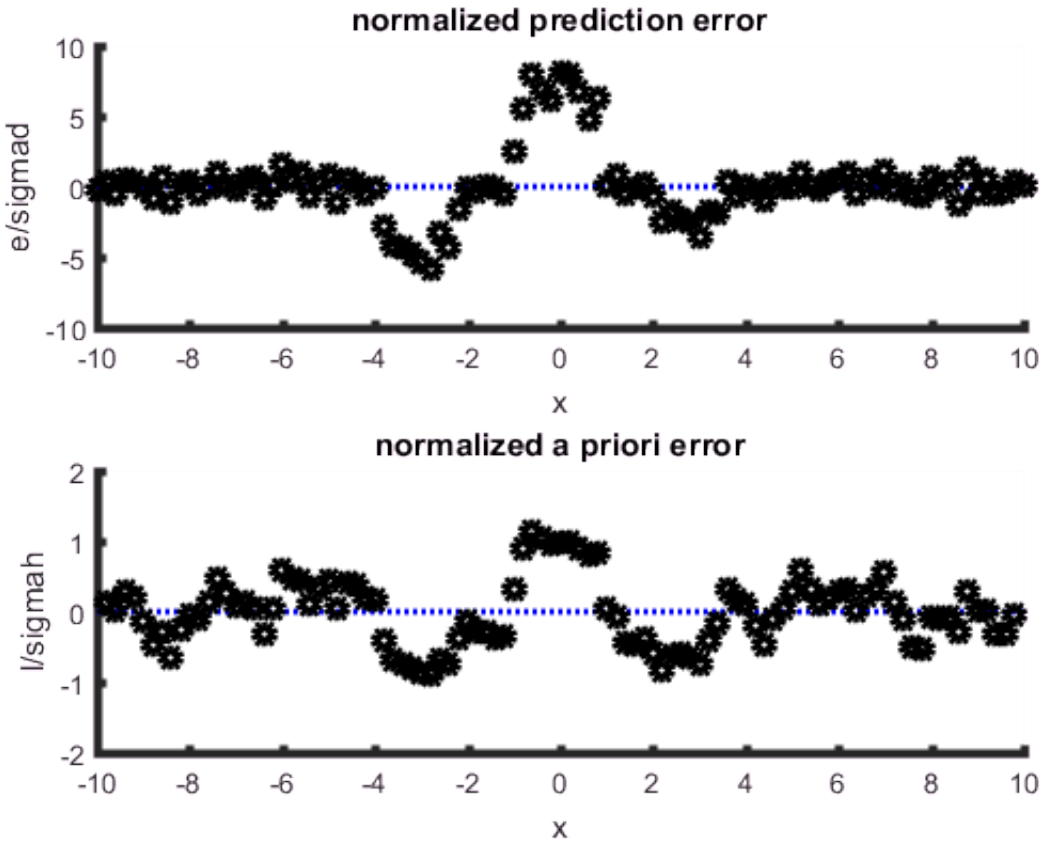
```
reslist=[1:floor(N/9):N];  % do these rows
Nres=length(reslist);
figure(3);
clf;
axis ij;
set(gca,'LineWidth', 3);
hold on;
axis( [xmin, xmax, xmin-(xmax-xmin)/Nres, xmax+(xmax-xmin)/Nres] );
xlabel('x');
ylabel('x');
title('resolution matrix');
vard = sigma2d * ones(N,1);  % variance of data
% (the 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 35 to a solution with relative residual 9.4e-06.
bicg converged at iteration 37 to a solution with relative residual 9.2e-06.
bicg converged at iteration 54 to a solution with relative residual 9.1e-06.
bicg converged at iteration 74 to a solution with relative residual 9.4e-06.
bicg converged at iteration 76 to a solution with relative residual 8.7e-06.
bicg converged at iteration 77 to a solution with relative residual 6.3e-06.
bicg converged at iteration 69 to a solution with relative residual 9.9e-06.
bicg converged at iteration 43 to a solution with relative residual 7.8e-06.
bicg converged at iteration 37 to a solution with relative residual 8.7e-06.
bicg converged at iteration 34 to a solution with relative residual 8.3e-06.
```



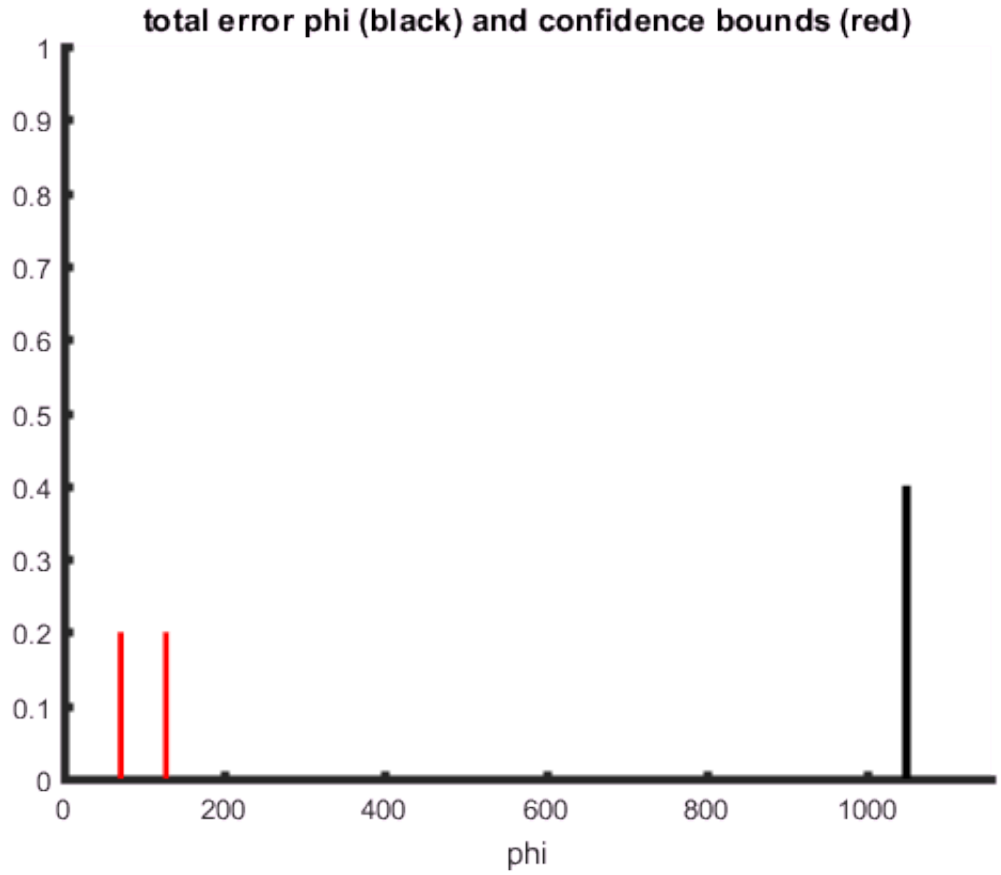

```
% Figure 3. Selected rows of the model resolution matrix.  They
% are peaaked along the main diagonal (good) but have small negative
% sidelobes (bad).

% Examine the individual errors

% 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, -10, 10] );
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 ormalized prediction error. Note that it is
% much larger than unity, and not uniform with x, indicating that
% that the model is not fitting the data very well. (Bottom) The
% model is fitting the priori information is a little better, since
% the size of the normalized error is closer to unity.  However, it
% is not not uniform in x, being largest around x=0. Our assumptions
% on the size and uniformity of sigmah is not quite right.

% 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.

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 total error phiest (black) is way above the 95%
% confidence interval (red bars), indicating that the high total
% error is unlikely to be due to random variation

% We have only one model, so a F-Test is not relevant
```