

```

% gda09_11
% Newton's method applied to inverse problem
% d(x)=g(x, m1, m2) with g = sin(w0*m(1)*x) + m(1)*m(2)
% supports Figure 9.9

clear all;

% data are in a single auxillary variable, x
N=40;
xmin=0;
xmax=1.0;
Dx=(xmax-xmin)/(N-1);
x = Dx*[0:N-1]';

% true model parameters
M=2;
mt = [1.21, 1.54]';

% d(x)=g(x, m1, m2) with g=sin(w0*m(1)*x) + m(1)*m(2)
w0=20;
dtrue = sin(w0*mt(1)*x) + mt(1)*mt(2); % true data
sd=0.4;
dobs = dtrue + random('Normal',0,sd,N,1); % observed data

% plot data
figure(1);
clf;
set(gca, 'LineWidth',3);
set(gca, 'FontSize',14);
hold on;
axis( [0, xmax, 0, 4] );
plot(x,dtrue,'k-','LineWidth',3);
plot(x,dobs,'ko','LineWidth',3);
xlabel('x');
ylabel('d');

% 2D grid, for plotting purposes only
L = 101;
Dm = 0.02;
m1min=0;
m2min=0;
m1a = m1min+Dm*[0:L-1]';
m2a = m2min+Dm*[0:L-1]';
m1max = m1a(L);
m2max = m2a(L);

% tabulate error, E, on grid for plotting purposed only
E = zeros(L,L);
for j = [1:L]
for k = [1:L]
    dpre = sin(w0*m1a(j)*x) + m1a(j)*m2a(k);
    E(j,k) = (dobs-dpre)'*(dobs-dpre);
end
end

% plot error surface
figure(2);
clf;
set(gca, 'LineWidth',3);

```

```

set(gca, 'FontSize', 14);
colormap('jet');
hold on;
axis( [m2min, m2max, m1min, m1max] );
axis ij;
imagesc( [m2min, m2max], [m1min, m1max], E);
colorbar;
xlabel('m2');
ylabel('m1');
plot( mt(2), mt(1), 'go', 'LineWidth', 3 );

% Newton's method, calculate derivatives
% y = sin( w0 m1 x ) + m1 m2;
% dy/dm1 = w0 x cos( w0 m1 x ) + m2
% dy/dm2 = m2

% initial guess and corresponding error
mg=[1,1]';
dg = sin(w0*mg(1)*x) + mg(1)*mg(2);
Eg = (dobs-dg)'*(dobs-dg);
plot( mg(2), mg(1), 'ko', 'LineWidth', 3 );

% save solution and minimum error as a function of iteration number
Niter=20;
Ehis=zeros(Niter+1,1);
m1his=zeros(Niter+1,1);
m2his=zeros(Niter+1,1);
Ehis(1)=Eg;
m1his(1)=mg(1);
m2his(1)=mg(2);

% iterate to improve initial guess
G = zeros(N,M);
for k = [1:Niter]

    % predict data at current guess for solution
    dg = sin( w0*mg(1)*x ) + mg(1)*mg(2);
    dd = dobs-dg;
    Eg=dd'*dd; % current error
    Ehis(k+1)=Eg; % save error history

    % build linearized data kernel
    G = zeros(N,2);
    G(:,1) = w0 * x .* cos( w0 * mg(1) * x ) + mg(2);
    G(:,2) = mg(2)*ones(N,1);

    % least squares solution for improvement to m
    dm = (G'*G)\(G'*dd);

    % update solution
    mg = mg+dm;
    plot( mg(2), mg(1), 'wo', 'LineWidth', 2 );

    % save solution history
    m1his(k+1)=mg(1);
    m2his(k+1)=mg(2);
    plot( [m2his(1:k-1), m2his(1+k) ], [m1his(1:k-1), m1his(1+k) ], 'r', 'LineWidth', 2 );

end

% estimated solution is from final iteration

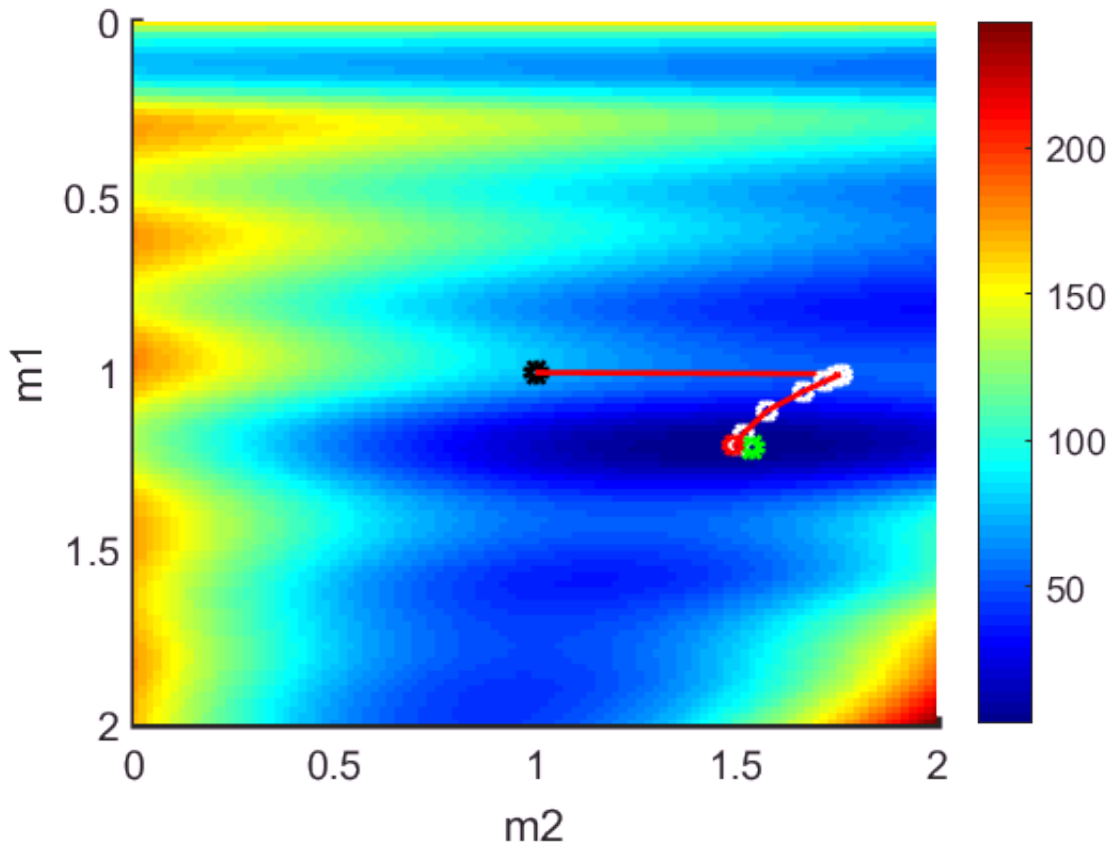
```

```

mlest = mlhis(Niter+1);
m2est = m2his(Niter+1);

% plot true and starting solution on error surface
plot( mt(2), mt(1), 'go', 'LineWidth', 2 );
plot( mg(2), mg(1), 'ro', 'LineWidth', 2 );

```



% Figure 9.9 Newton's method (linearized least squares) is used to solve the same  
 % nonlinear curve-fitting problem as in Figure 9.5. (B) Error  
 % surface (colors), showing true solution (green dot), and a series of improved  
 % solutions (white circles connected by red lines) determined by the method.

```

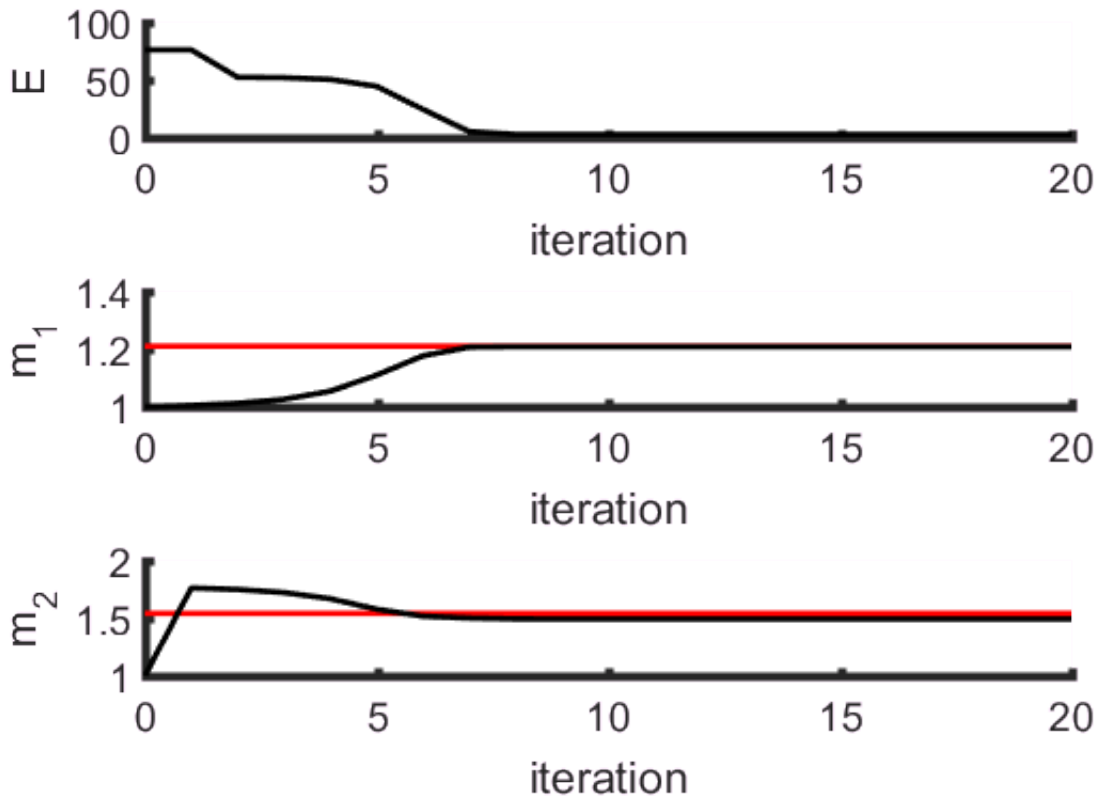
% plots of the history of E, m1 and m2
figure(3);
clf;
subplot(3,1,1);
set(gca,'LineWidth',3);
set(gca,'FontSize',14);
hold on;
plot( [0:Niter], Ehis, 'k-', 'LineWidth', 2 );
xlabel('iteration');
ylabel('E');
subplot(3,1,2);
set(gca,'LineWidth',3);
set(gca,'FontSize',14);
hold on;
plot( [0, Niter], [mt(1), mt(1)], 'r', 'LineWidth', 2 );
plot( [0:Niter], mlhis, 'k-', 'LineWidth', 2 );
xlabel('iteration');
ylabel('m_1');
subplot(3,1,3);

```

```

set(gca, 'LineWidth', 3);
set(gca, 'FontSize', 14);
hold on;
plot( [0, Niter], [mt(2), mt(2)], 'r', 'LineWidth', 2 );
plot( [0:Niter], m2his, 'k-', 'LineWidth', 2 );
xlabel('iteration');
ylabel('m_2');

```

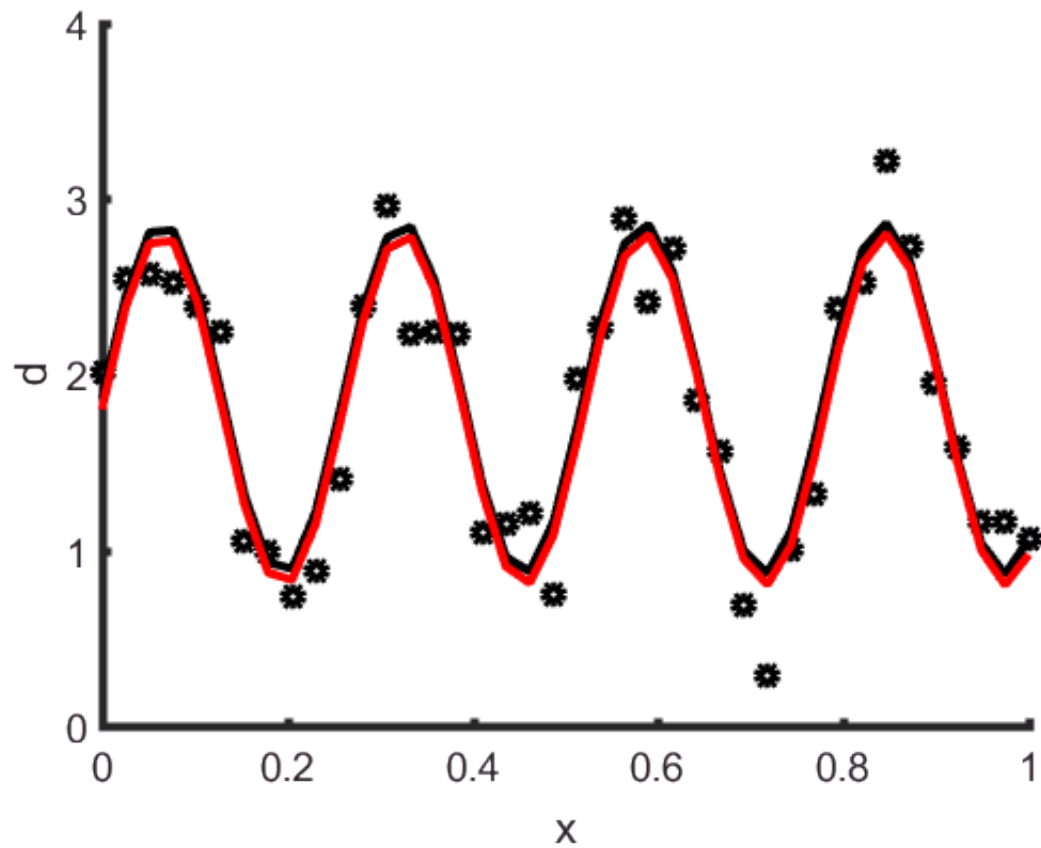


% Figure 9.9 Newton's method (linearized least squares) is used to solve the same  
 % nonlinear curve-fitting problem as in Figure 9.5. (C) Plot of error E and model  
 % parameters, m1 and m2 as a function of iteration number.

```

% evaluate prediction and plot it
figure(1);
dpre = sin(w0*mlest*x) + mlest*m2est;
plot(x,dpre,'r-','LineWidth',3);

```



% Figure 9.9 Newton's method (linearized least squares) is used to solve the same  
 % nonlinear curve-fitting problem as in Figure 9.5. (A) The observed data (black  
 % circles) are computed from the true data (black curve) by adding random noise.  
 % The predicted data (red curve) are based on the results of the method.