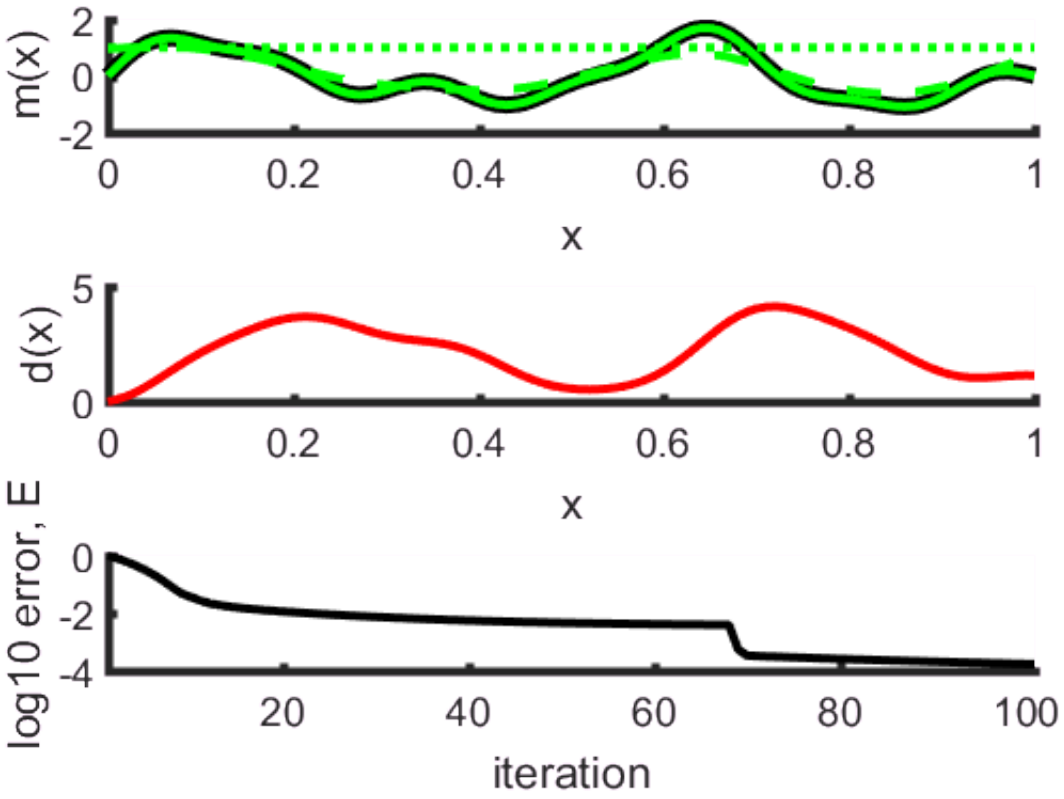
```
% gda11_03
% supports Figure 11.4
% gradient descent inversion of d(x)=Lm(x),
% where L is a linear operator. The gradient of
% error dE/dm is calculated using the adjoint method
% The linear operator is 0.1*(D + 2*I) where D is a
% first derivative and I is an integral

clear all;

% auxiliary variable x
M=101;
N=M;
Dx=1;
x = Dx*[0:M-1]'/(M-1);
xmax=max(x);

% a true model function m(x)that is zero at the boundaries
mtrue = sin(4*pi*x/xmax)+0.5*sin(7*pi*x/xmax)+0.25*sin(13*pi*x/xmax);

% plot the true model parameters
figure(1);
clf;
subplot(3,1,1);
hold on;
set(gca,'LineWidth',3);
set(gca,'FontSize',14);
plot(x,mtrue,'k-','LineWidth',6);
xlabel('x');
ylabel('m(x)');

% derivative and integral operators
D = (1/Dx)*toeplitz( [1, -1, zeros(1,M-2)]', [1, zeros(1,M-1)] );
I = Dx*toeplitz( ones(M,1), [1, zeros(1,M-1)] );

% data is a linear operator L on the model parameters, d=Lm
% where the linear operator contains both a derivative and an intergal:
% d = 0.1*( dm/dx + 2 * (integral 0 x) m dm )
L = 0.1*(D + 2*I);
dtrue = L*mtrue;

% plot the data
subplot(3,1,2);
set(gca,'LineWidth',3);
set(gca,'FontSize',14);
hold on;
plot(x,dtrue,'r-','LineWidth',3);
xlabel('x');
ylabel('d(x)');

% brute force dE/dm, to check
% mo = ones(M,1);
% do = L*mo;
% Eo = (dtrue-do)'*(dtrue-do);
% dEdm = zeros(M,1);
% Dm=0.00001;
% for i=[1:M]
%     dm = zeros(M,1);
%     dm(i)=Dm;
```

```
%     m1 = mo+dm;
%     d1 = L*m1;
%     E1 = (dtrue-d1)'*(dtrue-d1);
%     dEdm(i) = (E1-Eo)/Dm;
% end
% dEdmA = -2*(L'*(dtrue-do));
% fprintf('dEdm\n' );
% for i=[1:M]
%     fprintf('%d %f %f\n', i, dEdm(i), dEdmA(i) );
% end

% trial solution
mgo = ones(M,1);
subplot(3,1,1);
plot(x,mgo,'g:','LineWidth',3);

% maximum iterations
Niter=100000;

% save error history
Ehist = zeros(Niter+1,1);

% error and its gradient at the trial solution
dgo = L*mgo;
Ego = (dtrue-dgo)'*(dtrue-dgo);
dEdmo = -2*Dx*(L'*(dtrue-dgo));
tau=0.5;
c1=0.0001;
alpha=0.9;
Ehist(1) = Ego;

% gradient descent method
for k = [1:Niter]

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

    % backstep
    for kk=[1:10]
        mg = mgo+alpha*v;
        dg = L*mg;
        Eg = (dtrue-dg)'*(dtrue-dg);
        dEdm= -2*(L'*(dtrue-dg));
        if( (Eg<=(Ego + c1*alpha*v'*dEdmo)) )
            break;
        end
        alpha = tau*alpha;
    end

    % change in solution
    Dmg = sqrt( (mg-mgo)'*(mg-mgo) );

    % update
    mgo=mg;
    dgo = dg;
    Ego = Eg;
    dEdmo = dEdm;
    Ehist(k+1) = Ego;

    % plot one intermediate result
    if( k==40 )
       subplot(3,1,1);
```

```
        plot(x,mgo,'g--','LineWidth',3);
    end

    % terminate on small change
    if( Dmg < 1.0e-6 )
        break;
    end
end

fprintf('%d iterations\n',k);
```

```
15890 iterations
```

```
% plot result
subplot(3,1,1);
set(gca,'LineWidth',3);
set(gca,'FontSize',14);
plot(x,mgo,'g-','LineWidth',3);

% plot error history
subplot(3,1,3);
set(gca,'LineWidth',3);
set(gca,'FontSize',14);
hold on;
axis( [1, 101, -4, 0 ] );
plot([1:101],log10(Ehist(1:101)/Ehist(1)),'k-','LineWidth',3);
xlabel('iteration');
ylabel('log10 error, E');
```




```
% Figure 11.5 Example of the solution of a continuous inverse problem using a gradient method
```

```
% minimize the error E, where an adjoint method is used to compute ∇E. (A) A test function, m
% (green), trial function (dotted green) reconstructed function after 40 iterations (dashed gr
% and final reconstructed function after 15,890 iterations (green). (B) The data, d(t), satisf
% d(t) = ℒm(t), where ℒ is the linear operator discussed in the text. (C) Error E as a function
% iteration number, for the first 100 iterations. MatLab script gda11_03.
```