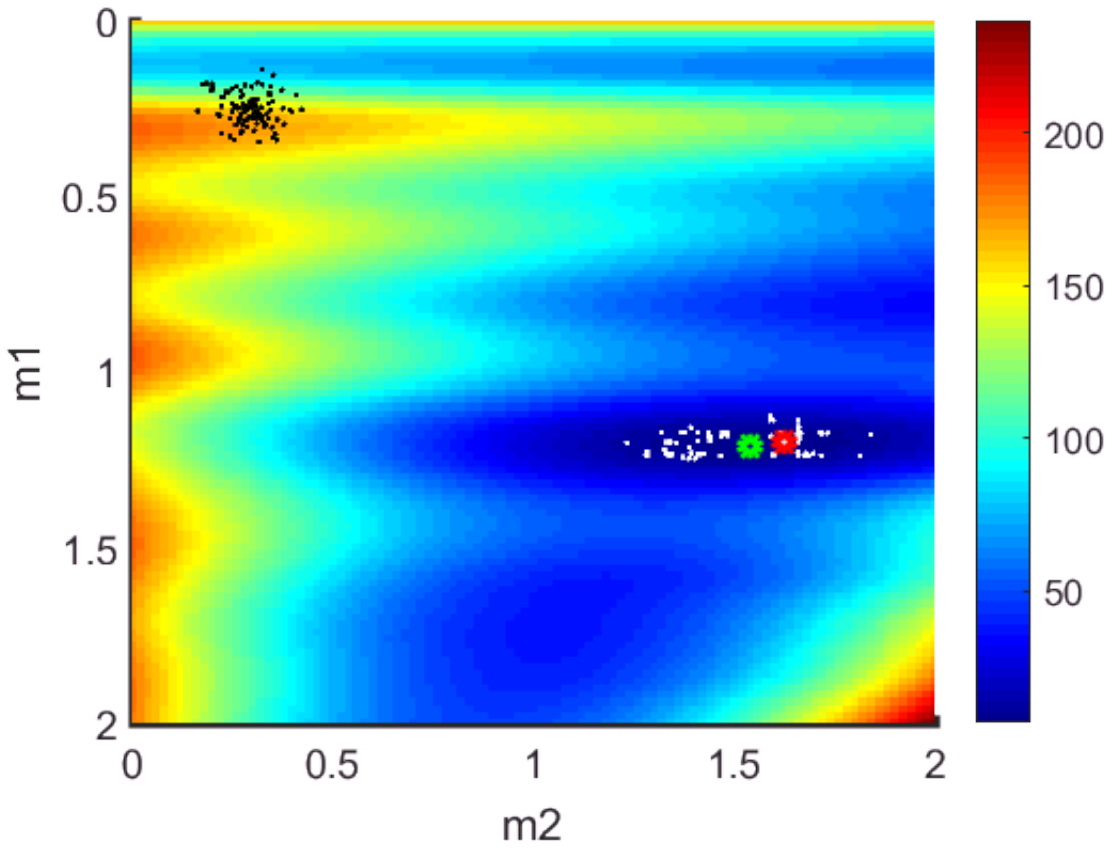
```
% gda09_18
% Genetic Algorithm applied to inverse problem
% d(x)=g(x, m1, m2) with g = sin(w0*m(1)*x) + m(1)*m(2)
% Supports Figure 9.16

clear all;

global gray1 gray1index pow2;
load('gray1table.mat');  % load table of Gray codes

% two model parameters
M=2;

% Number of generations
L=40;
Esave = zeros(L,1);
Msave = zeros(L, M);

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

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

% y=f(x, m1, m2);
w0=20;
dtrue = sin(w0*mt(1)*x) + mt(1)*mt(2);
sd=0.4;
dobs = dtrue + random('Normal',0,sd,N,1);

% 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');

% define 2D grid
% this is just for plotting purposes
% the genetic algorith does not require a grid
Lm = 101;
Dm = 0.02;
m1min=0;
m2min=0;
m1a = m1min+Dm*[0:Lm-1]';
m2a = m2min+Dm*[0:Lm-1]';
m1max = m1a(Lm);
m2max = m2a(Lm);

% compute error E on the grid
```

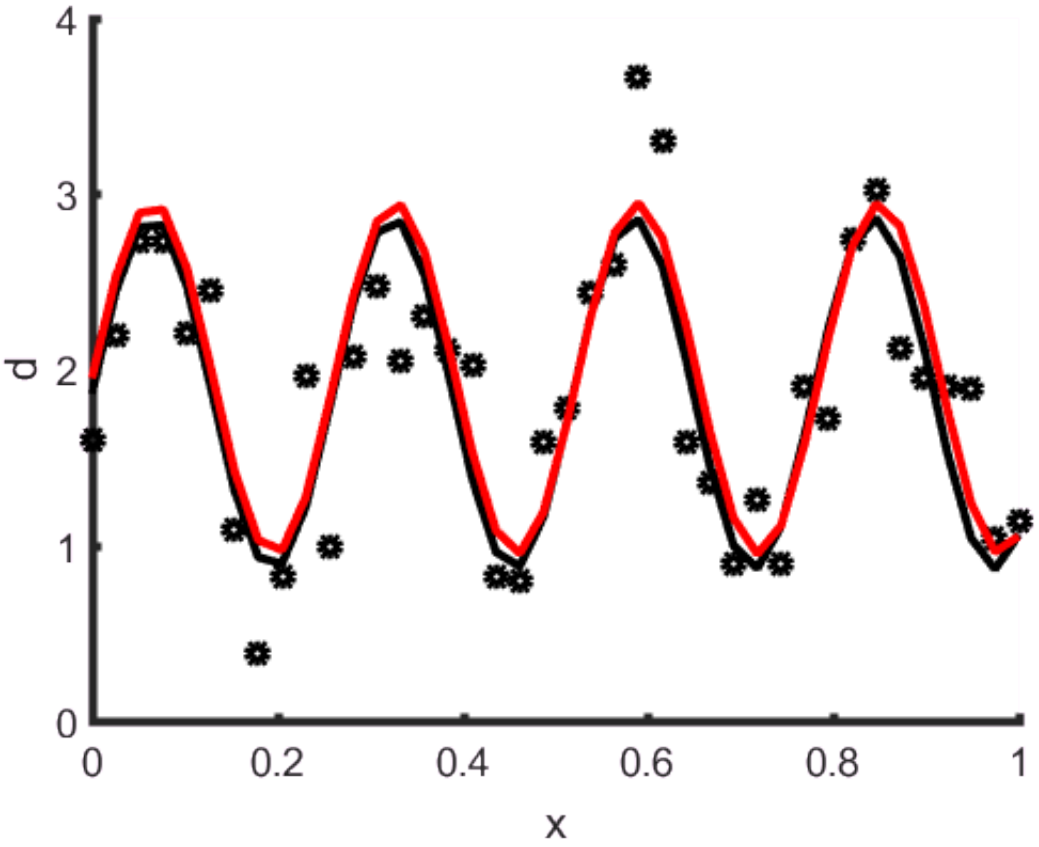
```
% this is just for plotting purposes
% the genetic algorith does not require a grid
E = zeros(Lm,Lm);
for j = [1:Lm]
for k = [1:Lm]
    dpre = sin(w0*m1a(j)*x) + m1a(j)*m2a(k);
    E(j,k) = (dobs-dpre)'*(dobs-dpre);
end
end

K = 100;  % number of individuals in the population
Kold = K;
m0 = [0.25, 0.30]; % initial guess for m1
mL = [m1min, m2min]'; % solution > mL
mR = [m1max, m2max]'; % solution < mR

dpre = sin(w0*m0(1)*x) + m0(1)*m0(2); % initial prediction
e = dobs-dpre; % individual errors
E0 = e'*e; % total error at start

% figure 2 is the error surface
figure(2);
clf;
set(gca,'LineWidth',3);
hold on;
set(gca,'FontSize',14);
colormap('jet');
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 );

mind = zeros(K, M); % solution for each individual
Eind = E0*ones(K,1); % error for each individual
F = ones(K,1); % fitness of each individual
genes = char(48*ones(K, 16*M)); % genes for each individual
for i=[1:M]
    r = random('Normal',0,0.05,K,1);
    mind(:,i) = m0(i)*ones(K,1)+r; % all individuals initialized to initial guess
end
for i=[1:K]
for j=[1:M]
    k = floor(65535.999*(mind(i,j)-mL(j))/(mR(j)-mL(j)));
    k1 = (j-1)*16+1;
    k2 = k1+15;
    genes(i,k1:k2) = gda_int2gray(k);
end
end
plot( mind(:,2), mind(:,1), 'k.', 'LineWidth', 2 );

for gen=[1:L]

% each individual replicates
mind = [mind;mind];
genes = [genes;genes];
Eind = [Eind;Eind];
K = 2*K;
```

```
% mutate genes
for i=[1:K]
    if( unidrnd(2,1) == 1 ) % mutate only half of individuals
        continue;
    end
    k = unidrnd(16*M,1); % one random mutation
    if( genes(i,k)=='0' )
         genes(i,k)='1';
    else
         genes(i,k)='0';
    end
end

% conjugate genes; only one conjugation per generation
j1 = unidrnd(K,1); % random individual
j2 = unidrnd(K,1); % another random individual
k = unidrnd(M*16-2,1)+1;  % 2<=k<=16M-1
g1(1,1:16*M) = genes(j1,:);
g1(1,k+1:16*M) = genes(j2,k+1:16*M);
g2(1,1:16*M) = genes(j2,:);
g2(1,k+1:16*M) = genes(j1,k+1:16*M);
genes(j1,:) = g1;
genes(j2,:) = g2;

% update solution and error from genes
for i=[1:K]
for j=[1:M]
    k1 = (j-1)*16+1;
    k2 = k1+15;
    k = gda_gray2int(genes(i,k1:k2));
    mind(i,j) = (mR(j)-mL(j))*(k/65535.999)+mL(j);
end
end
for i=[1:K]
    dpre = sin(w0*mind(i,1)*x) + mind(i,1)*mind(i,2);
    e = dobs-dpre;
    Eind(i) = e'*e;
end
Emax = max(Eind);

% survival of the fitness
c=5;
F = exp(-c*Eind/Emax).*random('uniform',0,1,K,1);
[F_sorted, k] = sort(F);
k = k(Kold+1:2*Kold,1);
K = Kold;
mind = mind(k,1:M);
Eind = Eind(k,1);
genes = genes(k,1:16*M);

[Emin, k] = min(Eind);
Esave(gen,1) = Emin;
msave(gen,1:M)= mind(k,1:M);

end

plot( mind(:,2), mind(:,1), 'w.', 'LineWidth', 2 );
plot( mt(2), mt(1), 'go', 'LineWidth', 3 );

% evaluate prediction and plot it
[Emin, k] = min(Eind);
```

```
plot( mind(k,2), mind(k,1), 'ro', 'LineWidth', 3 );
```

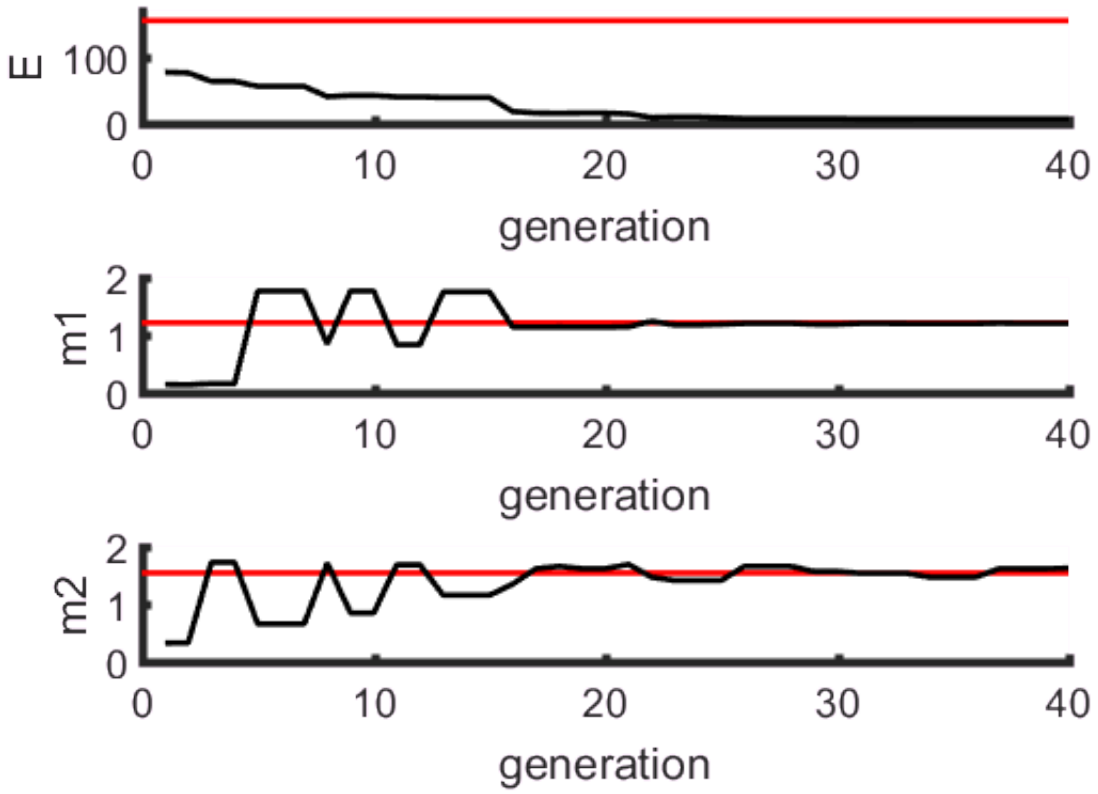



```
% Figure 9.16 (B) Error surface (colors), showing true solution (green
% circle), the initial population of solutions (black dots), the population of
% solutions after 40 generations and the best estimate solution (red circle).

figure(1);
dpre = sin(w0*mind(k,1)*x) + mind(k,1)*mind(k,2);
plot(x,dpre,'r-','LineWidth',3);
```




```
% Figure 9.16 The genetic algorithm 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.

% figure 3 is the solution progress
figure(3);
clf;
subplot(3,1,1); % error
set(gca,'LineWidth',3);
set(gca,'FontSize',14);
hold on;
axis( [0, L, 0, 1.1*E0] );
plot( [0, L]', [E0, E0]', 'r-', 'LineWidth', 2);
plot( [1:L]', Esave, 'k-', 'LineWidth', 2);
xlabel('generation');
ylabel('E');
subplot(3,1,2); % m1
set(gca,'LineWidth',3);
set(gca,'FontSize',14);
hold on;
axis( [0, L, 0, 2] );
plot( [0, L]', [mt(1), mt(1)]', 'r-', 'LineWidth', 2);
plot( [1:L]', msave(:,1)', 'k-', 'LineWidth', 2);
xlabel('generation');
ylabel('m1');
subplot(3,1,3); % m2
set(gca,'LineWidth',3);
set(gca,'FontSize',14);
hold on;
```

```
axis( [0, L, 0, 2] );
plot( [0, L]', [mt(2), mt(2)]', 'r-', 'LineWidth', 2);
plot( [1:L]', msave(:,2)', 'k-', 'LineWidth', 2);
xlabel('generation');
ylabel('m2');
```




```
% Figure 9.16 The genetic algorithm is used to solve the same nonlinear curve-fitting
% problem as in Figure 9.5. (C) Plot of error Eand model parameters m_1 and m_2  as a
% function of generation. MatLab script gda09_18.
```