```
% gda09_avg
% Genetic algorithm applied to an "approximately
% separable inverse problem" where the M=20 model parameters
% have a natural ordering and the data are localized
% averages of the model parameters.

% This code compares error of two cases, A and B,
% where the probability of mutation and recombination
% differ between the cases.  Each case has 5 trials.

% Supports Figure 9.17

clear all;
global gray1 gray1index pow2;
load('gray1table.mat');

% model parameters
M=20;

% Number of generations
L=50;
Esave = zeros(L,1);
Msave = zeros(L, M);

% plot error
figure(1);
clf;
set(gca,'LineWidth',3);
set(gca,'FontSize',14);
hold on;
axis( [1, L, 0, 1.5 ] );
xlabel('generation');
ylabel('E');

Ntrials=5;

% part 1 ------------------------------------------
Nrecom = 0;
imutate = 2;

for itrial=[1:Ntrials]

% number of recombinations per generation


% fitness coefficient
c=5;

% true model parameters are linear ramp
mt = 0.5+[0:M-1]'/(M-1);

% data average of 3 neighboring model parameters
N=M;
x = [1:M]';
dtrue = (mt+[0;mt(1:M-1)]+[0;0;mt(1:M-2)])/3;
sd=0.025;
dobs = dtrue + random('Normal',0,sd,N,1);

K = 100;  % number of individuals in the population
```

```
Kold = K;
m0 = ones(M,1); % initial guess for m1
mL = zeros(M,1); % solution > mL
mR = 2*ones(M,1)'; % solution < mR

mind = zeros(K, M); % solution for each individual
Eind = zeros(K,1); % error for each individual
Ftind = zeros(K,1); % fitness of each individual
genes = char(48*ones(K, 16*M)); % genes for each individual
% all individuals initialized to initial guess + random number
for i=[1:M]
    r = random('Normal',0,0.025,K,1);
    mind(:,i) = m0(i)*ones(K,1)+r;
end
% compute initial error
for i=[1:K]
    dpre = (mind(i,:)'+[0;mind(i,1:M-1)']+[0;0;mind(i,1:M-2)'])/3; % predicetd data
    e = dobs-dpre;
    Eind(i) = e'*e;
end
Emax = max(Eind);
Emaxstart = Emax;
Ftind = exp(-c*Eind/Emax).*random('uniform',0,1,K,1); % initial fitness
for i=[1:K] % convert model parameters to genes
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

for gen=[1:L] % loop over generations

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

% mutate genes
for i=[1:K]
    if( unidrnd(imutate,1) == 1 ) % mutate 1/imutate 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

% recominate genes; Nrecom per generation
for rc = [1:Nrecom]
    j1 = unidrnd(K,1); % random individual
    j2 = unidrnd(K,1); % another random individual
    k = 16*(unidrnd(M-2,1)+1)+1;  % always at start of word
    g1(1,1:16*M) = genes(j1,:);
    g1(1,k:16*M) = genes(j2,k:16*M);
    g2(1,1:16*M) = genes(j2,:);
```

```
    g2(1,k:16*M) = genes(j1,k:16*M);
    genes(j1,:) = g1;
    genes(j2,:) = g2;
end

% 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 = (mind(i,:)'+[0;mind(i,1:M-1)']+[0;0;mind(i,1:M-2)'])/3; % predicetd data
    e = dobs-dpre;
    Eind(i) = e'*e;
end
Emax = max(Eind);

% survival of the fitness
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

% evaluate prediction and plot it
[Emin, i] = min(Eind);
dpre = (mind(i,:)'+[0;mind(i,1:M-1)']+[0;0;mind(i,1:M-2)'])/3; % predicetd data

plot([1:L]',Esave,'k-','LineWidth',2);
fprintf('Done with A %d\n', itrial);
end
```

```
Done with A 1
Done with A 2
Done with A 3
Done with A 4
Done with A 5
```

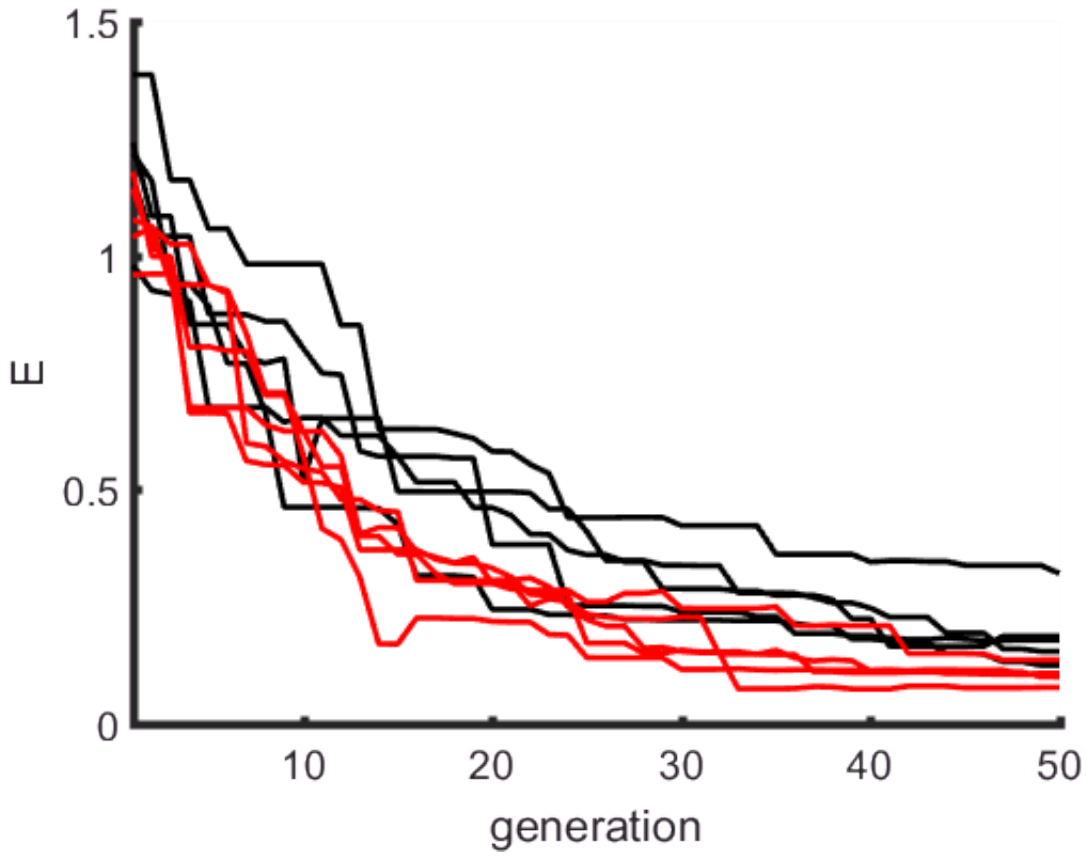
```
% part 2 ---------------------------------------------
Nrecom = 5;
imutate = 8;

for itrial=[1:Ntrials]

% fitness coefficient
c=5;
```

```
% true model parameters are linear ramp
mt = 0.5+[0:M-1]'/(M-1);

% data average of 3 neighboring model parameters
N=M;
x = [1:M]';
dtrue = (mt+[0;mt(1:M-1)]+[0;0;mt(1:M-2)])/3;
sd=0.025;
dobs = dtrue + random('Normal',0,sd,N,1);

K = 100;  % number of individuals in the population
Kold = K;
m0 = ones(M,1); % initial guess for m1
mL = zeros(M,1); % solution > mL
mR = 2*ones(M,1)'; % solution < mR

mind = zeros(K, M); % solution for each individual
Eind = zeros(K,1); % error for each individual
F = zeros(K,1); % fitness of each individual
genes = char(48*ones(K, 16*M)); % genes for each individual
% all individuals initialized to initial guess + random number
for i=[1:M]
    r = random('Normal',0,0.025,K,1);
    mind(:,i) = m0(i)*ones(K,1)+r;
end
% compute initial error
for i=[1:K]
    dpre = (mind(i,:)'+[0;mind(i,1:M-1)']+[0;0;mind(i,1:M-2)'])/3; % predicetd data
    e = dobs-dpre;
    Eind(i) = e'*e;
end
Emax = max(Eind);
Emaxstart = Emax;
F = exp(-c*Eind/Emax).*random('uniform',0,1,K,1); % initial fitness
for i=[1:K] % convert model parameters to genes
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

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(imutate,1) == 1 ) % mutate 1/imutate 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

% recominate genes; Nrecom per generation
for rc = [1:Nrecom]
    j1 = unidrnd(K,1); % random individual
    j2 = unidrnd(K,1); % another random individual
    k = 16*(unidrnd(M-2,1)+1)+1;  % always at start of word
    g1(1,1:16*M) = genes(j1,:);
    g1(1,k:16*M) = genes(j2,k:16*M);
    g2(1,1:16*M) = genes(j2,:);
    g2(1,k:16*M) = genes(j1,k:16*M);
    genes(j1,:) = g1;
    genes(j2,:) = g2;
end

% 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 = (mind(i,:)'+[0;mind(i,1:M-1)']+[0;0;mind(i,1:M-2)'])/3; % predicetd data
    e = dobs-dpre;
    Eind(i) = e'*e;
end
Emax = max(Eind);

% survival of the fitness
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

fprintf('Done with B %d\n', itrial);

% evaluate prediction and plot it
[Emin, i] = min(Eind);
dpre = (mind(i,:)'+[0;mind(i,1:M-1)']+[0;0;mind(i,1:M-2)'])/3; % predicetd data

plot([1:L]',Esave,'r-','LineWidth',2);
end
```

```
Done with B 1
Done with B 2
Done with B 3
Done with B 4
```

```
Done with B 5
```




```
% Figure 9.17 Genetic algorithm applied to the approximately-separable localized average =
% problem discussed in the text. The minimum error E_i^min error in a population of 100,
% declines with generation i. The rate of decline is faster for a combination of evolutionary
% parameters that includes recombination (red curves, 12.5% mutation rate, 5% recombination
% rate) than for one that omits it (black curves,  50% mutation, 0% recombination). Histories
% of five populations are shown for each parameter set.  MatLab script gda09_19
```