**Homework 3**

**Problem restatement:**  
Consider a function , where x and y follow the table below.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| x | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| y | 0 | 0.4560 | 0.7614 | 0.8586 | 0.7445 | 0.4661 | 0.1045 | -0.2472 | -0.5073 | -0.6233 | -0.5816 |

1. Find the optimal parameters by applying the Monte Carlo Method and the gradient descent method, and start the iterations from an initial guess .

Theory basis in the gradient descent method:

In Levenberg Method, the iteration formula is as follows,

(1-1)

If we have a very large , then we can nearly ignore the 2nd derivative of E, and the iteration formula is as follows,

(1-2)

where .

Also, we need Armijo’s rule to provide an acceptance criterion for , which is given as follows.

(1-3)

where, .

In this problem, the matrix is ,

(1-4)

After calculation, the optimal parameters are: , and the comparison between the observed data and the predicted data is as figure 1 shows.

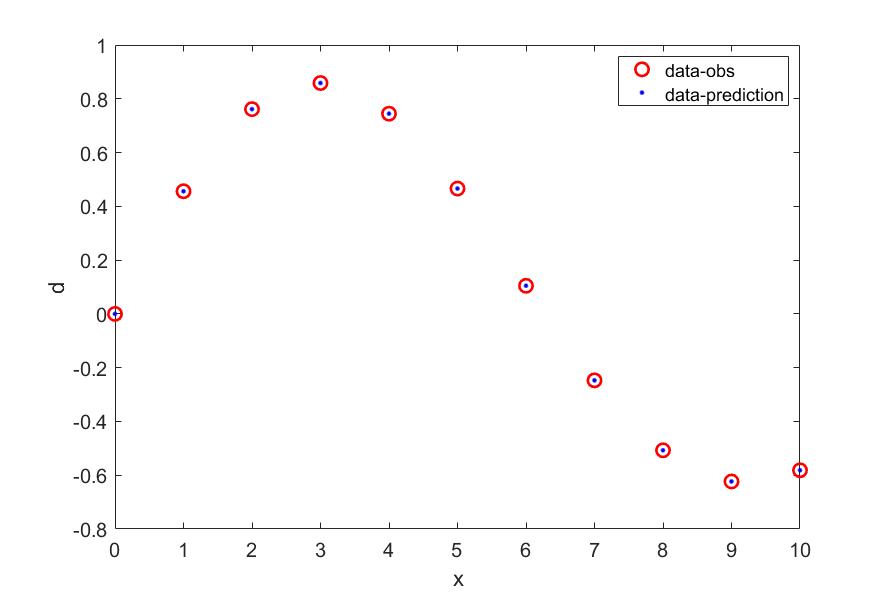


figure 1. Comparison between the observed and predicted data by gradient descent method

1. The Monte Carlo Method is a method based on probability, which computes the error at randomly generated points in model space. In this problem, the model space is [0,1]. As figure 2 shows, in Monte Carlo Method, when the iteration number is 10000, the optimal parameter is, , and the corresponding figure is below which compares the observation data and the predicting data. The result shows inversion result does not fit better than Gradient method.

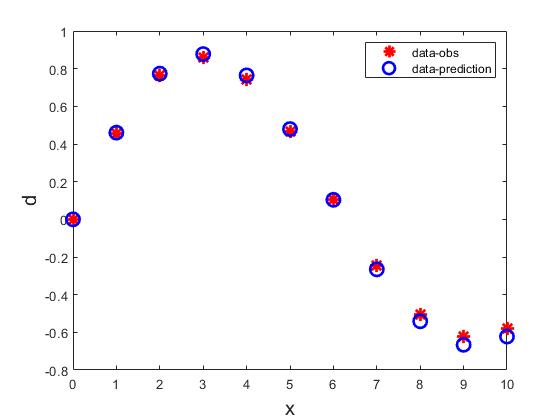


figure 2. Comparison between the observed and predicted data by Monte Carlo method

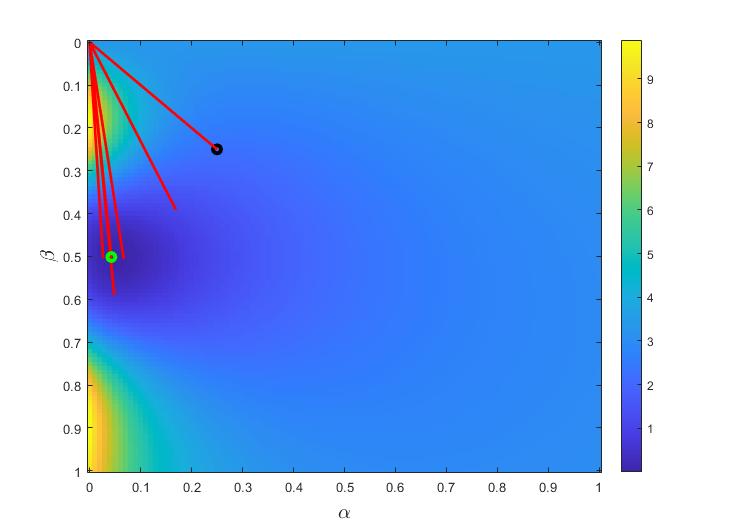


figure 3. Iteration path of Monte Carlo method

1. Change the initial guess via grid search within an initial model space . Draw the series of improved solutions for each initial model in an error surface.

First, we draw the error space with a range of zero to two by step 0.02, and the colormap is as follows, then, we set different initial models by black marking point, and the global minimum is set by green marking point. As figure 4 and figure 5 show, all the initial models we choose in the searching space can yield the global minimum(The iterative method is gradient descent method).

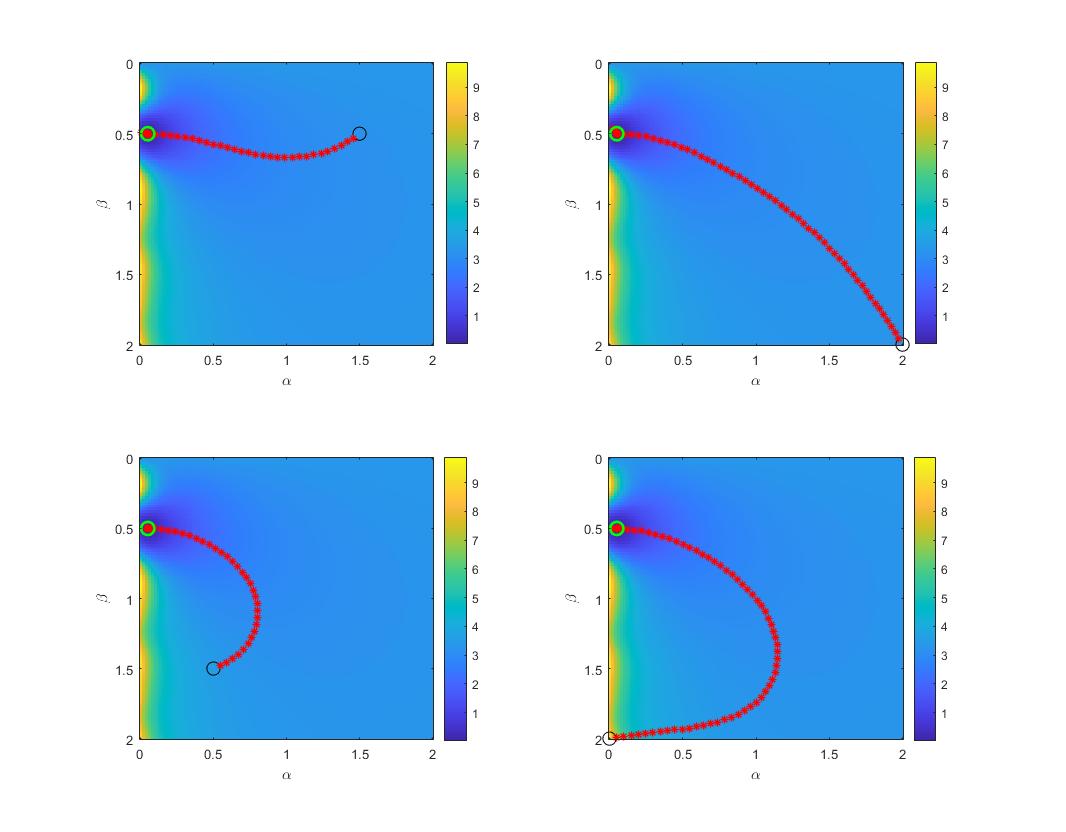


figure 4. Series of improved solutions of four initial models

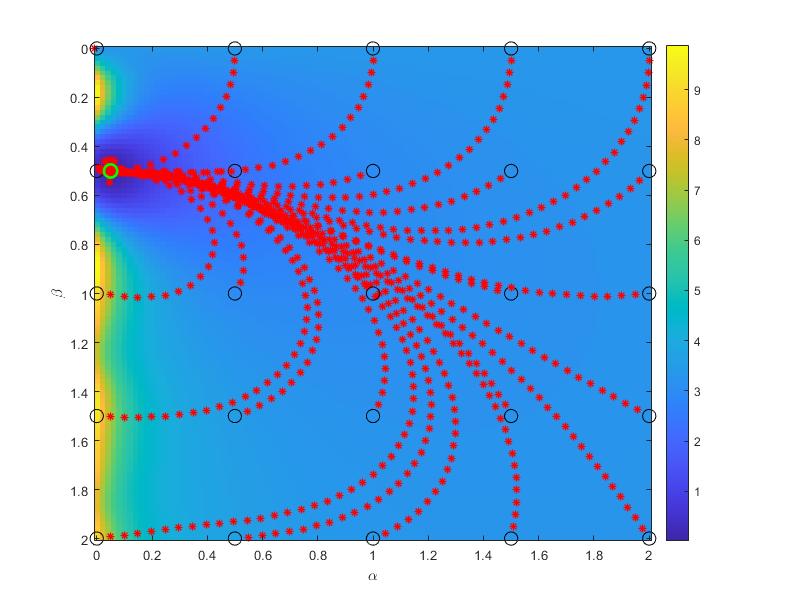


figure 5. Series of improved solutions of various initial models

**Conclusions**

Comparing the gradient descent method and the Monte Carlo method, the first method can give better performance both in calculating speed and the accuracy. But when meeting complex situations, the gradient method can only give a local minimum, not mentioned in this problem.

Matlab code:

%% GRADIENT DESCENT METHOD

clear;

clc;

x = (0:1:10)';

N = length(x);

mgo = [0.25,0.25]';%give the initial guess of alpha and beta

ygo = exp(-mgo(1)\*x).\*sin(mgo(2)\*x);

dobs = [0,0.4560,0.7614,0.8586,0.7445,0.4661,0.1045,-0.2472,-0.5073,-0.6233,-0.5816]';

Ego = (ygo-dobs)'\*(ygo-dobs);

% calculate the Gp

dydmo = zeros(N,2);

dydmo(:,1) = -mgo(1)\*exp(-mgo(1)\*x).\*sin(mgo(2)\*x);

dydmo(:,2) = mgo(2)\*exp(-mgo(1)\*x).\*cos(mgo(2)\*x);

dEdmo = 2\*dydmo'\*(ygo-dobs);% calculate the vector b

alpha = 0.05;c1 = 0.0001;tau = 0.5;Niter = 500;

for k = 1:Niter

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

for kk = 1:10

mg = mgo + alpha\*v; % add the value of the step number to update the mg

yg = exp(-mg(1)\*x).\*sin(mg(2)\*x);

Eg = (yg - dobs)'\*(yg - dobs);

dydm = zeros(N,2);

dydm(:,1) = -mg(1)\*exp(-mg(1)\*x).\*sin(mg(2)\*x);

dydm(:,2) = mg(2)\*exp(-mg(1)\*x).\*cos(mg(2)\*x);

dEdm = 2\*dydm'\*(yg - dobs);

if(Eg <= (Ego + c1\*alpha\*v'\*dEdmo))

break;

end

alpha = tau\*alpha;

end

Dmg = sqrt((mg - mgo)'\*(mg - mgo));

mgo = mg;ygo = yg; Ego = Eg;

dydmo = dydm;dEdmo = dEdm;

if(Dmg < 1.0e-6)

break;

end

end

fprintf("The optimal parameters: alpha = %.2f, beta = %.2f\n",mg(1),mg(2));

figure(1)

plot(x,dobs,'ro',x,yg,'b.','LineWidth',2,'MarkerSize',10),xlabel('x','FontSize',15),ylabel('d','FontSize',15),legend('data-obs','data-prediction')

%% MONTE CARLO METHOD

clear;

clc;

m=[0:0.01:1;0:0.01:1]';

x=(0:10)';

dobs=[0,0.456,0.7614,0.8596,0.7445,0.4661,0.1045,-0.2472,-0.5073,-0.6233,-0.5816]';

E=zeros(101,101);

for i=1:101

for j=1:101

yg=exp(-m(i,1)\*x).\*sin(m(j,2)\*x);

Eg1=abs((yg-dobs)'\*(yg-dobs));

E(i,j)=Eg1;

end

end

figure(1)

imagesc(0:0.01:1,0:0.01:1,E');

colorbar;

mg = [0.25,0.25]';

dg = exp(-mg(1)\*x).\*sin(mg(2)\*x);

Eg = (dobs - dg)'\*(dobs - dg);

hold on

plot(mg(1),mg(2),'ko',"LineWidth",3);

%randomly generate pairs of model parameters and check if they further

%minimize the error

Niter = 10000;

ma = zeros(2,1);

choosek = [100,200,300,1000,2000,3000,4000,5000,6000,7000,8000,9000];

for k = 1:Niter

ma(1) = random('Unif',0,1);

ma(2) = random('Unif',0,1);

%compute the error

da = exp(-ma(1)\*x).\*sin(ma(2)\*x);

Ea = (dobs - da)'\*(dobs-da);

%adopt it if it is better

if (Ea < Eg)

mg = ma;

Eg = Ea;

end

%save history

Ehis = zeros(length(k),1);

m1his = zeros(length(k),1);

m2his = zeros(length(k),1);

Ehis(1+k) = Eg;

m1his(1+k) = mg(1);

m2his(1+k) = mg(2);

hold on

plot([m1his(1+k-1),m1his(1+k)],[m2his(1+k-1),m2his(1+k)],'r','linewidth',2)

fprintf('已经执行%d次\n',k);

end

hold on

plot(mg(1),mg(2),'go','LineWidth',3);

xlabel('\alpha','FontSize',15),ylabel('\beta','FontSize',15)

figure(2)

dg1 = exp(-mg(1)\*x).\*sin(mg(2)\*x);

plot(x,dobs,'r\*',x,dg1,'bo','LineWidth',2,'MarkerSize',10),xlabel('x','FontSize',15),ylabel('d','FontSize',15),legend('data-obs','data-prediction')

% grid search method

clear;

clc;

m=[0:0.02:2;0:0.02:2]';

x=(0:10)';

dobs=[0,0.456,0.7614,0.8596,0.7445,0.4661,0.1045,-0.2472,-0.5073,-0.6233,-0.5816]';

E=zeros(101,101);

for i=1:101

for j=1:101

yg=exp(-m(i,1)\*x).\*sin(m(j,2)\*x);

Eg=(yg-dobs)'\*(yg-dobs);

E(i,j)=Eg;

end

end

imagesc(0:0.02:2,0:0.02:2,E');

colorbar

xlabel('\alpha');

ylabel('\beta');

mgo=zeros(2,1);

N=length(x);

ma=[0:0.5:2;0:0.5:2]';% set different initial guess

n=length(ma);

for i=1:n

for j=1:n

mgo(1)=ma(i,1);

mgo(2)=ma(j,1);

hold on;

plot(mgo(1),mgo(2),'ko','MarkerSize',10);

ygo=exp(-mgo(1)\*x).\*sin(mgo(2)\*x);

Ego=(ygo-dobs)'\*(ygo-dobs);

dydmo=zeros(N,2);

dydmo(:,1)=-x.\*exp(-mgo(1)\*x).\*sin(mgo(2)\*x);

dydmo(:,2)=exp(-mgo(1)\*x).\*cos(mgo(2)\*x).\*x;

dEdmo=2\*dydmo'\*(ygo-dobs);

alpha = 0.05;c1=0.0001;tau=0.5;Niter=500;

for k=1:Niter

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

for kk=1:10

mg=mgo+alpha\*v;

hold on

plot(mg(1),mg(2),'r\*','LineWidth',1);

yg=exp(-mg(1)\*x).\*sin(mg(2)\*x);

Eg=(yg-dobs)'\*(yg-dobs);

dydm=zeros(N,2);

dydm(:,1)=-x.\*exp(-mg(1)\*x).\*sin(mg(2)\*x);

dydm(:,2)=exp(-mg(1)\*x).\*cos(mg(2)\*x).\*x;

dEdm=2\*dydm'\*(yg-dobs);

if((Eg<=(Ego+c1\*alpha\*v'\*dEdmo)))

break;

end

alpha=tau\*alpha;

end

Dmg=sqrt((mg-mgo)'\*(mg-mgo));

mgo=mg;

ygo=yg;

Ego=Eg;

dydmo=dydm;

dEdmo=dEdm;

if(Dmg<1.0e-6)

break;

end

end

end

end

hold on

plot(mg(1),mg(2),'go','MarkerSize',10,'LineWidth',2)