**Homework II**

Consider the data in the table below,

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **y** | 0.0250 | 0.0750 | 0.1250 | 0.1750 | 0.2250 | 0.2750 | 0.3250 | 0.3750 | 0.4250 | 0.4750 |
| **d(y)** | 0.2388 | 0.2319 | 0.2252 | 0.2188 | 0.2126 | 0.2066 | 0.2008 | 0.1952 | 0.1898 | 0.1846 |
| **y** | 0.5250 | 0.5750 | 0.6250 | 0.6750 | 0.7250 | 0.7750 | 0.8250 | 0.8750 | 0.9250 | 0.9750 |
| **d(y)** | 0.1795 | 0.1746 | 0.1699 | 0.1654 | 0.1610 | 0.1567 | 0.1526 | 0.1486 | 0.1447 | 0.1410 |

1. Using the data provided, we can discretize the integral equation to create a square G matrix.

(1-1)

which can be written as follows,

(1-2)

To expand the equation, we have,

*...*

(1-3)

The length of is 20, and for G is a square matrix, the size of G is , and the equations can be written as follows,

(1-4)

(1-5)

To discretize the x, we construct the same vector as y, which is from 0.025 to 0.975, and the interval is 0.05.

Firstly, we use to calculate vector **m**, we get the plot of m(i) as figure 1, and then we use to calculate **m**, we get figure 2, finally we take both of them back to **,** and then we get figure 3, showing that the result of calculation can fit the original data well.

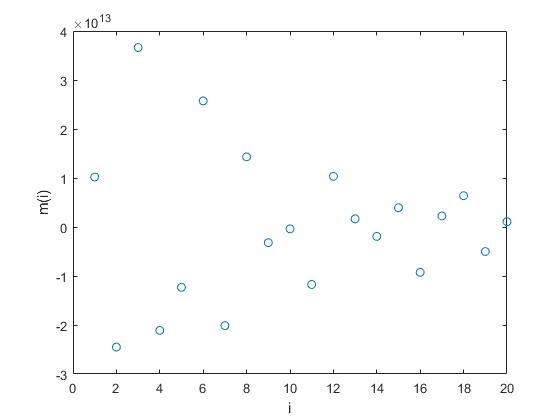


fig.1 Plot of m(i)

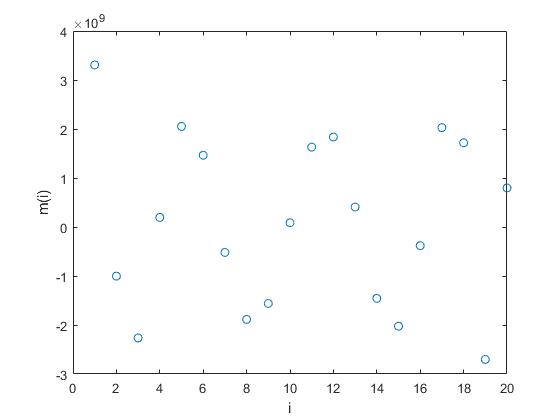


fig.2 Plot of m(i)

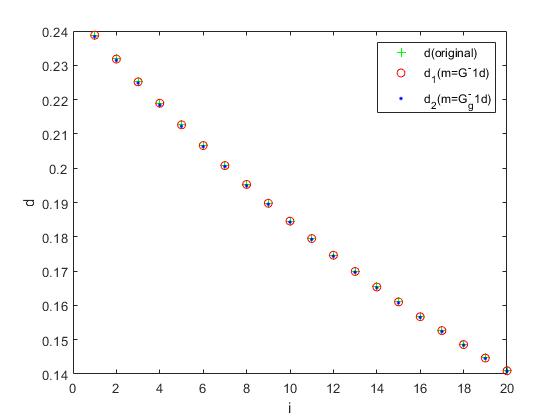


fig.3 The calculated result

1. We can use the command, [U,S,V] = svd(G), to accomplish the singular value decomposition, and the parameter S will return as the singular value matrix as follows, and the size of S is 20\*20, the diagonal elements of S matrix are the singular values we want, which are 0.4184, 0.0238, 7.0589\*10(-4),...,2.7693\*10(-19).

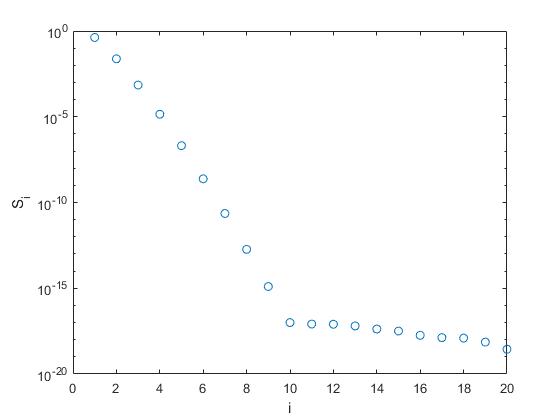


fig.4. Singular values of G

1. The condition number of G is or , is the singular number of G, and we use the command **cond(G,2)** to calculate, and the result is , so the matrix G is ill-posed.
2. For the natural solution, the discrete Picard condition is satisfied when the numerator decays more quickly than the denominator, and the solution is as follows.

(2-1)

The discrete Picard condition plot is as figure 5 shows.

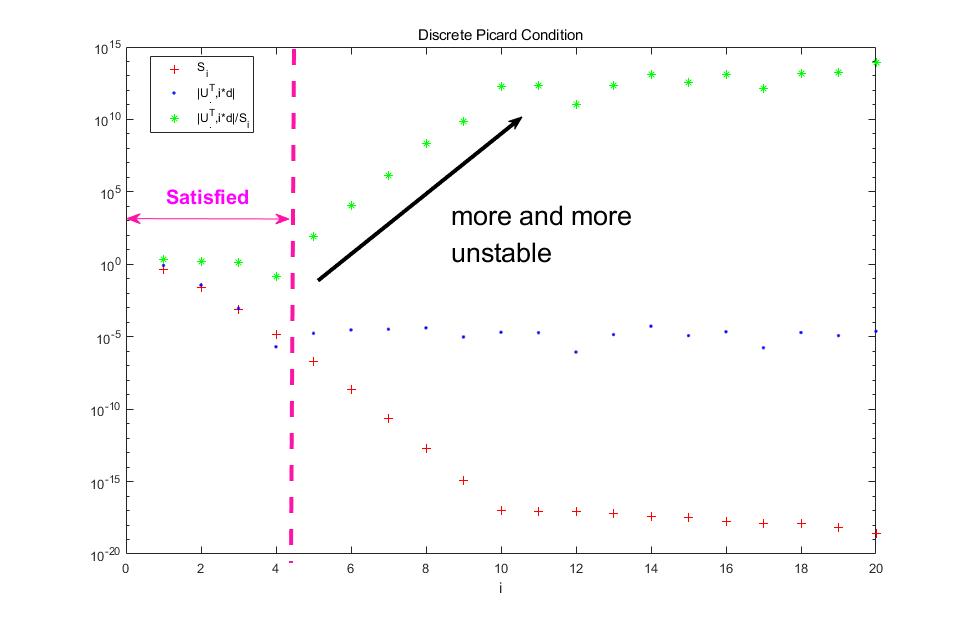


fig.5 Discrete Picard Condition

As figure 5 shows, we take the first 4 singular values to use for the TSVD method, then the calculated result can be shown in figure 6.

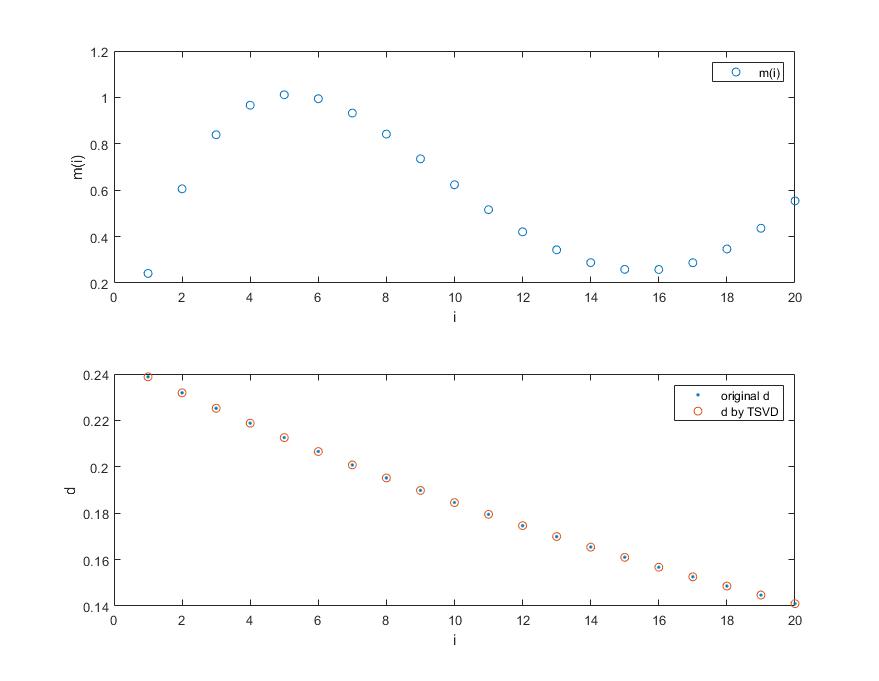


fig.6 The result of TSVD

From figure 6, we can see the TSVD behaves well, and the m(i) varies smoothly.

1. The Regularized SVD-form is as follows,

(3-1)

which can be written in matrix form,

(3-2)

let , we get figure 7 as follows, which compare the TSVD method with the SVD method.

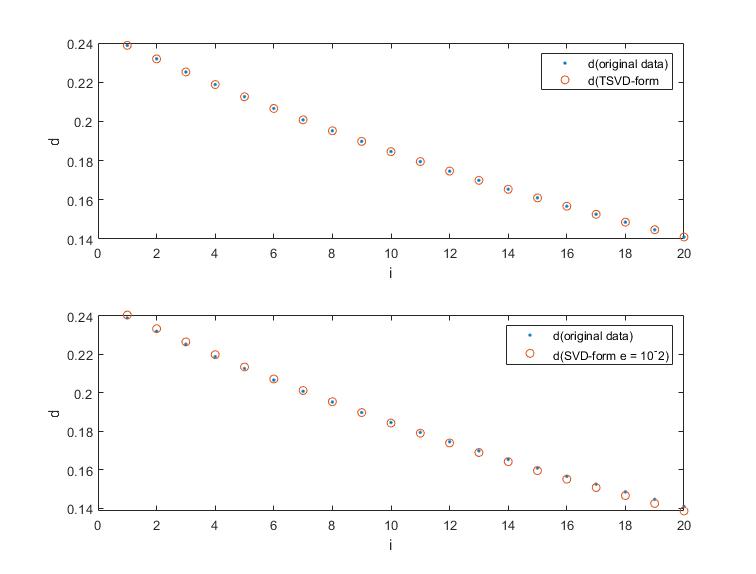


fig.7 SVD and TSVD result

From figure 7, we can see the value of can affect the fitting result.

To pick the proper regularization parameter, we use the L-curve criterion and GCV.

For L-curve criterion, we just need to calculate under various .

When plotted on a log-log scale, the curve of optimal values of versus often takes on a characteristic “L” shape in linear problems, which happens because is a strictly decreasing function of and is a strictly increasing function of , the L-curve plot is as figure 8 shows.

In figure 8, we can find the corner point, and plot this point onto the figure 9.

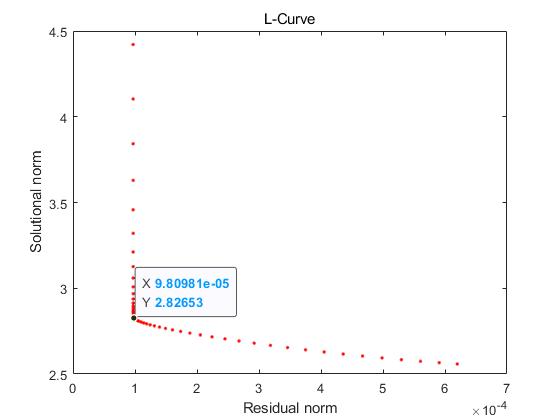


fig.8 L-Curve (I)

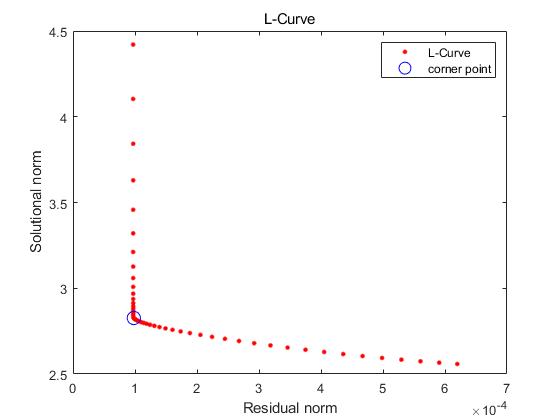


fig.9 L-Curve (II)

For Generalized Cross Validation(GCV), we have this equation,

(3-3)

Thus, we get the GCV curve below,

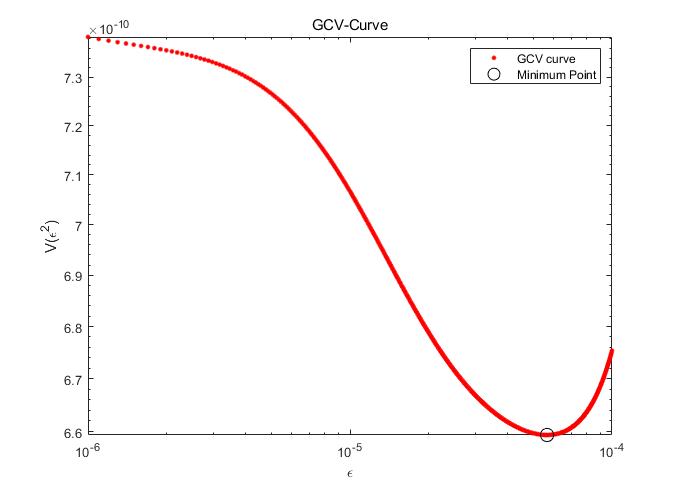


fig.10 GCV Curve

Above all, we have the we want, also called regularization parameter, and figure 11 shows the best fitting with L-Curve and GCV method.

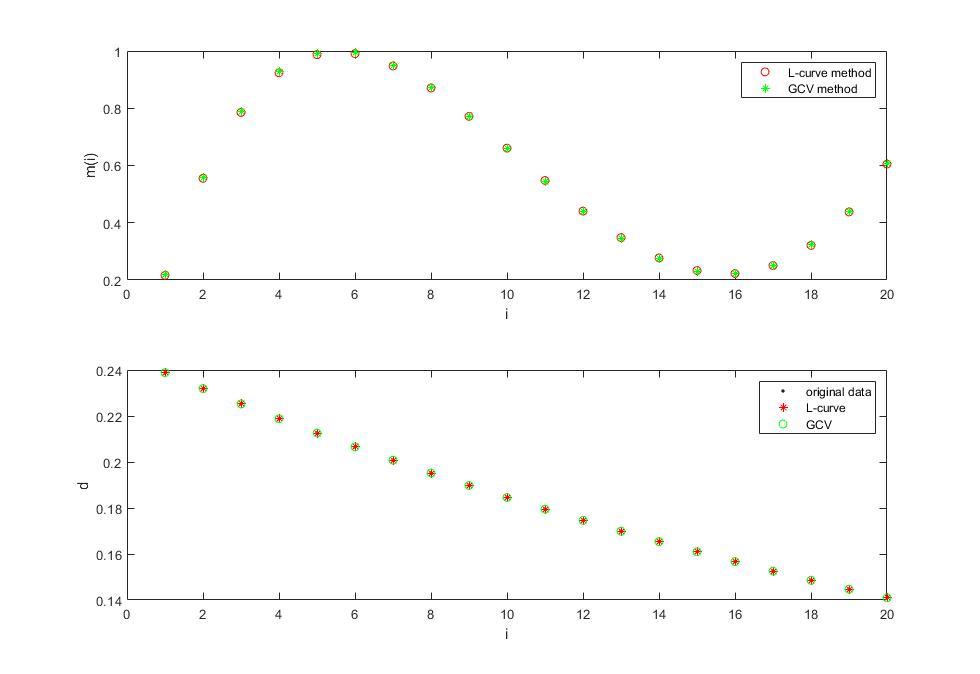


fig.11 Fitting result with L-curve method and GCV

**Appendix(MATLAB CODE):**

clear;

clc;

format shortE %important announcement, because the default accuracy in MATLAB is not enough

d = [0.2388,0.2319,0.2252,0.2188,0.2126,0.2066,0.2008,0.1952,0.1898,0.1846,0.1795,0.1746,0.1699,0.1654,0.1610,0.1567,0.1526,0.1486,0.1447,0.1410]';

y = (0.025:0.05:0.9750)';

% discretize the x

x = (0.025:0.05:0.9750)';

dx = 0.05;

% caculate G

G = zeros(20,length(x));

for a = 1:length(y)

for j = 1:length(x) %he column of G matrix is related to the length of x

G(a,j) = dx\*x(j)\*exp(-x(j)\*y(a));

end

end

%calculate the unknown m(x)

m = G\d;

m1 = pinv(G)\*d;

d1 = G\*m;

d2 = G\*m1;

z = (1:20);%number of node

figure(1)

plot(z,d,'g+',z,d1,'ro',z,d2,'b.'),legend('d(original)','d\_1(m=G^-1d)','d\_2(m=G\_g^-1d)')

xlabel('i')

ylabel('d')

figure(2)

plot(z,m,'o')

xlabel('i')

ylabel('m(i)')

figure(3)

plot(z,m1,'o')

xlabel('i')

ylabel('m(i)')

%to calculate the singular value and the condition number of G

[U,S,V] = svd(G);

Singular\_matrix = S;

condition\_number = cond(G,2);

%make the plot of picard condition

for a=1:20

Ud(a) = abs(U(:,a)'\*d);

end

Si = diag(S);

figure(8)

plot(z,Si,'o'),xlabel('i'),ylabel('S\_i'),set(gca,'Yscale','log')

for c = 1:20

Ud1(c) = Ud(c)/Si(c);

end

figure(4)

plot(z,Si,'r+',z,Ud,'b.',z,Ud1,'g\*'),legend('S\_i','|U^T\_.,i\*d|','|U^T\_.,i\*d|/S\_i')

set(gca,'Yscale','log')

title('Discrete Picard Condition')

xlabel('i')

set(gca,'yminortick','on')

% using TSVD

for ii = 1:4

S1(ii,ii) = S(ii,ii);

end

for jj = 1:4

V1(:,jj) = V(:,jj);

end

for kk = 1:4

U1(:,kk) = U(:,kk);

end

mest = V1\*inv(S1)\*U1'\*d;

figure(5)

subplot(211)

plot(z,mest,'o'),legend('m(i)')

xlabel('i'),ylabel('m(i)')

d3 = G\*mest;

subplot(212)

plot(z,d,'.',z,d3,'o'),legend('original d','d by TSVD')

xlabel('i'),ylabel('d')

% choose an arbitrary e to calculate

e = 10^(-2);

mest2 = inv(G'\*G+e^2\*I)\*G'\*d;

d4 = G\*mest2;

figure(9)

subplot(211)

plot(z,d,'.',z,d3,'o'),legend('d(original data)','d(TSVD-form)')

xlabel('i'),ylabel('d')

subplot(212)

plot(z,d,'.',z,d4,'o'),legend('d(original data)','d(SVD-form e = 10^-2)')

xlabel('i'),ylabel('d')

clear;

clc;

%initial parameter

dx=0.05;

p=1:20;

N=20;

x=(0.025:0.05:0.975);

y=(0.025:0.05:0.975);

d = [0.2388 0.2319 0.2252 0.2188 0.2126 0.2066 0.2008 0.1952 0.1898 0.1846 0.1795 0.1746 0.1699 0.1654 0.1610 0.1567 0.1526 0.1486 0.1447 0.1410]';

G=zeros(20,20); %initialize the G

% construct the G matrix

for j=1:length(y)

for i=1:length(x)

G(i,j)=dx\*x(j)\*exp(-x(j)\*y(i));

end

end

e = logspace(-6,-3,100);

SN = zeros(length(e),1);%to initialize the solutional norm

RN = zeros(length(e),1);%to initialize the residual norm

%calculate the 2-norm of the two parameters

for i=1:length(e)

ml = (G'\*G+e(i)^2\*eye(N))\(G'\*d);

dl = G\*ml;

SN(i) = norm(ml);

RN(i) = norm(dl-d);

end

corner = 65;

SNc = SN(corner);

RNc = RN(corner);

e1 = e(corner);

figure(2)

plot(RN,SN,'r.',RNc,SNc,'bo','MarkerSize',10),legend('L-Curve','corner point')

% plot(RN,SN,'r.')

% set(gca,'Yscale','log')

% set(gca,'Xscale','log')

xlabel('Residual norm');

ylabel('Solutional norm');

title('L-Curve');

%plot the GCV curve

e2 = linspace(1e-6,1e-4,1000);%has the better plot

V = zeros(length(e2),1);

for i=1:length(e2)

m2 = (G'\*G+e2(i)^2\*eye(N))\(G'\*d);

d2 = G\*m2;

V(i) = N\*((d-d2)'\*(d-d2)/(trace(eye(N)-G\*((G'\*G+e2(i)^2\*eye(N))\G')))^2);

end

[minimumV,point] = min(V);

egcv = e2(point);

figure(3);

loglog(e2,V,'r.',e2(point),minimumV,'ko','MarkerSize',10);

legend('GCV curve','Minimum Point');

xlabel('\epsilon');

ylabel('V(\epsilon^2)');

title('GCV-Curve');

%exammine the GCV and L-curve method

mL = (G'\*G + e1^2 \* eye(N))\(G'\*d);

dL = G\*mL;

mgcv = (G'\*G + egcv^2 \* eye(N))\(G'\*d);

dgcv = G \* mgcv;

figure(4);

subplot(2,1,1)

plot(p,mL,'ro',p,mgcv,'g\*'),xlabel('i'),ylabel('m(i)')

legend('L-curve method','GCV method')

subplot(2,1,2)

plot(p,d,'k.',p,dL,'r\*',p,dgcv,'go'),xlabel('i'),ylabel('d')

legend('original data','L-curve','GCV')