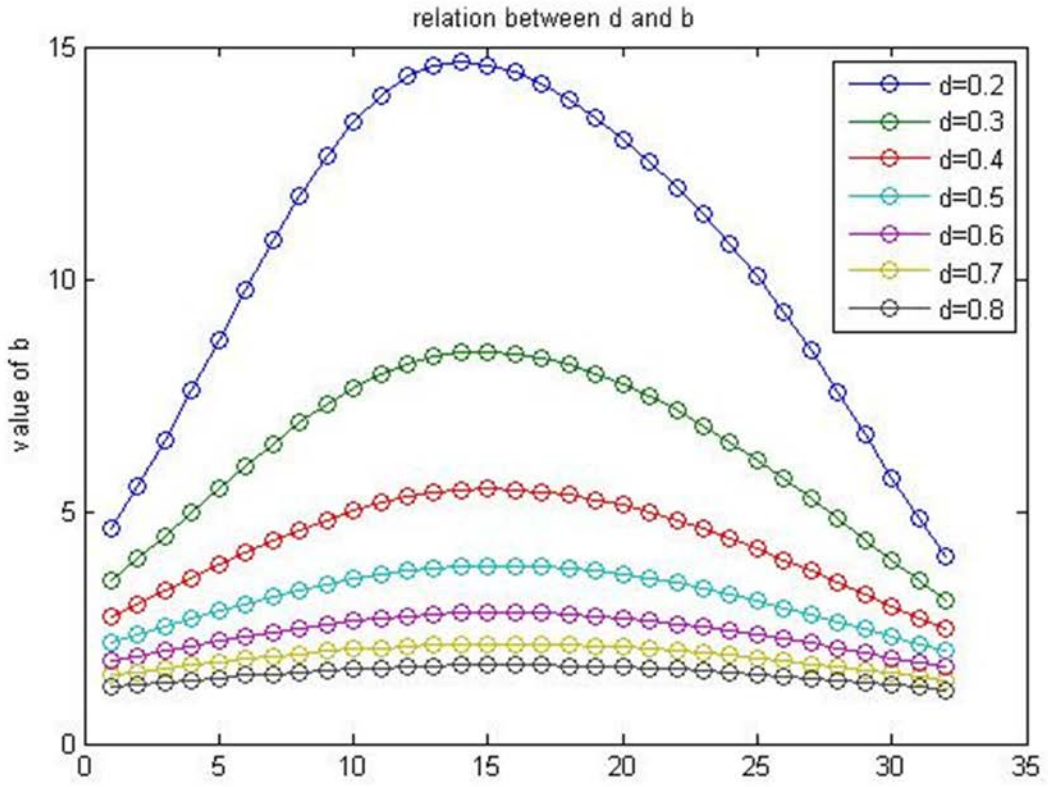
IP Project 1

Shuowen Wei

2012.2.13

Problem 1.

We let **d** vary from 0.2 to 0.8 and get the corresponding value of **b** showed in the graph below:



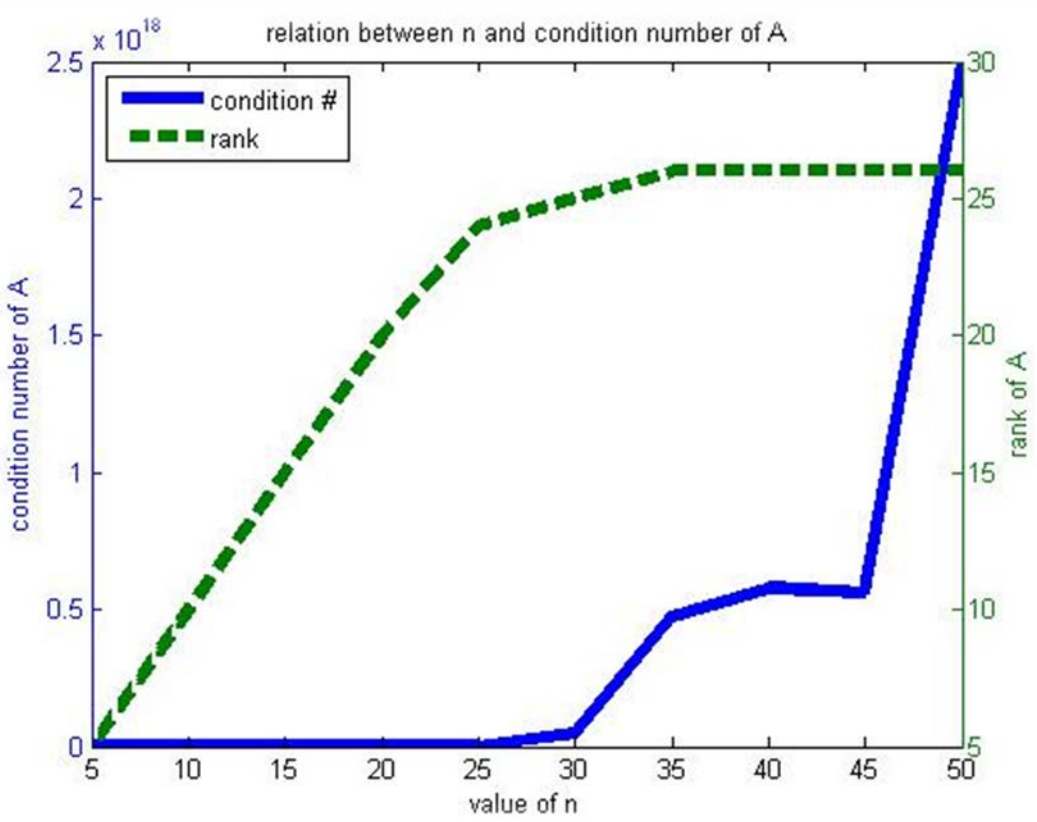
The table below shows the corresponding maximum and minimum value of **b** with respect to different depth **d**:

d	0.2	0.3	0.4	0.5	0.6	0.7	0.8
Max **b**	14.666	8.427	5.479	3.829	2.817	2.152	1.693
Min **b**	4.030	3.096	2.460	1.997	1.647	1.376	1.162

Obviously we can see that when **d** is larger, then the max and min value of **b** is smaller, and the curve of **b**, from the graph, becomes more smoother and narrower.

Problem 2.

We fix **d**=0.5, let **n** vary from 5 to 50 and get the corresponding condition number of matrix **A** in the graph below:



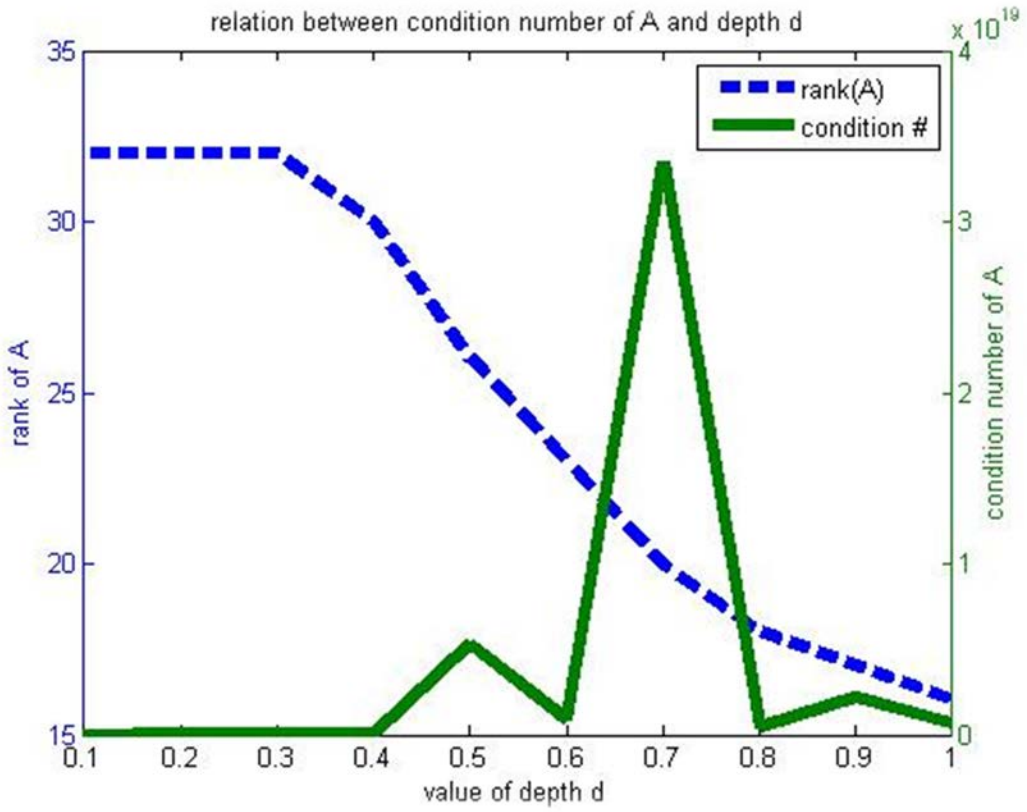
Even when **n**=5 very small, the condition number of matrix A is 93.74, very big. Thus, we also compute the rank of **A** according to each **n** and show them in the table below:

n	5	10	15	20	25
rank(**A**)	5	10	15	20	24
n	30	35	40	45	50
rank(**A**)	25	26	26	26	26

The output **A** should be n-by-n matrix and Toeplitz, but we can see that **A** becomes rank deficient when n=25 (or between 20 and 25, to be precise).

Problem 3.

Now, we keep **n** fixed at 32, and then let **d** vary from 0. 1 to 1 and compute the corresponding condition number of matrix **A** (which is n-by-n matrix), the results are showed in the graph below:



When the depth **d** is small, say less than 0.3, the matrix **A** is still full rank, i.e. rank(**A**) equals **n** which is fixed at 32, and we have the smallest value of the condition number of **A**, min(cond(**A**))=2524.04, which is also very large. When **d** is greater than 0.3, then **A** becomes rank deficient, and theoretically speaking, the condition number of a rank deficient matrix is ∞.

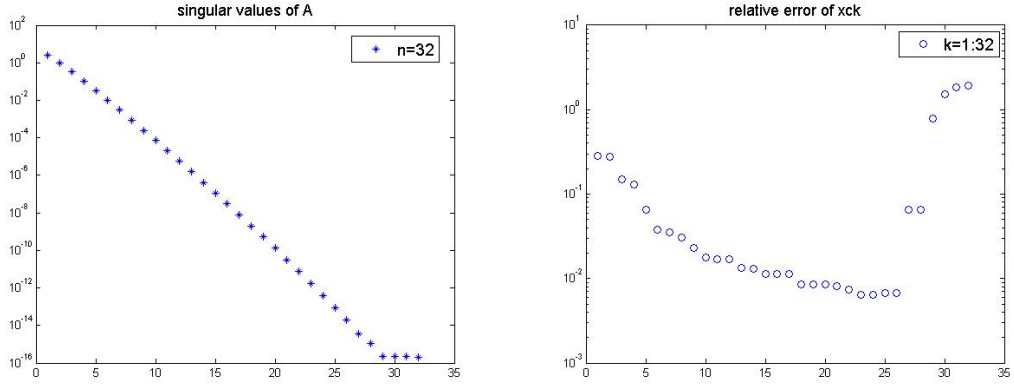
Problem 4.

We set **n**=32, **example**=2 and **d**=0.5, under the compact format in Matlab, by using the backslash commend "\" we get the relative error of **xc** is 12.8303.

This large error implies the result **xc** computed by ***backslash*** commend is very far away from the true solution **x**, this is because when **n**=32, **d**=0.5, based on the observations from problem 2 and 3, we already know that the output matrix **A** is rank deficient and with super large condition number, thus for this an ill-conditioned problem, directly computing the solution by ***backslash*** commend will make the result very unstable. In the coming problem 5 we will try to use the **truncated SVD** solve such ill-conditioned problem.

Problem 5.

Under the same set of the parameters with problem 4, we use ***semilogy*** to plot the singular values of matrix **A** below, from the left of the graph we can see how these singular values of **A** decay:



The **truncated SVD** ktsvd.m can be found in the appendix, and we tried every **k** from 1 to **n**=32 to find which one minimizes the relative error of **xck**, also showed by ***semilogy*** at the right of the graph above. It turned out be k=24 has the minimum relative error equals 0.006458.

When **k**=16, the relative error is 0.011396.

Problem 6.

(1).

We set **n**=32, **d**=0.5 and compute the solution for the normal equation by ***backslash*** and get the each relative error for each λ as follows:

λ	0.0001	0.001	0.01	0.1
relative error	0.0203	0.0314	0.0466	0.1249

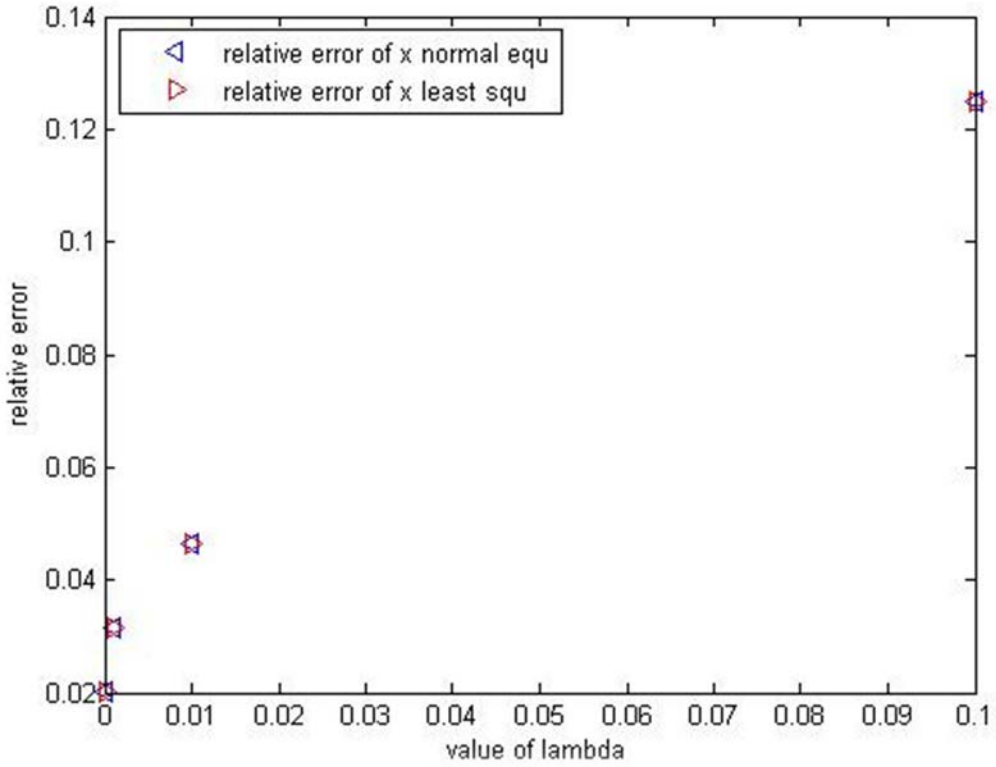
And obviously, $\lambda = 0.0001$ gives the smallest relative error in this solution method.

(2).

With the same parameter set as (1), we compute least squares problem by ***lsqlin*** and get the each relative error for each λ as follows:

λ	0.0001	0.001	0.01	0.1
relative error	0.0203	0.0314	0.0466	0.1249

And also, $\lambda = 0.0001$ gives the smallest relative error in this solution method. As a matter of fact, these two methods are equivalent.



Problem 7.

We set **n**=32, **d**=0.5 and **example**=2 for the whole problem 7:

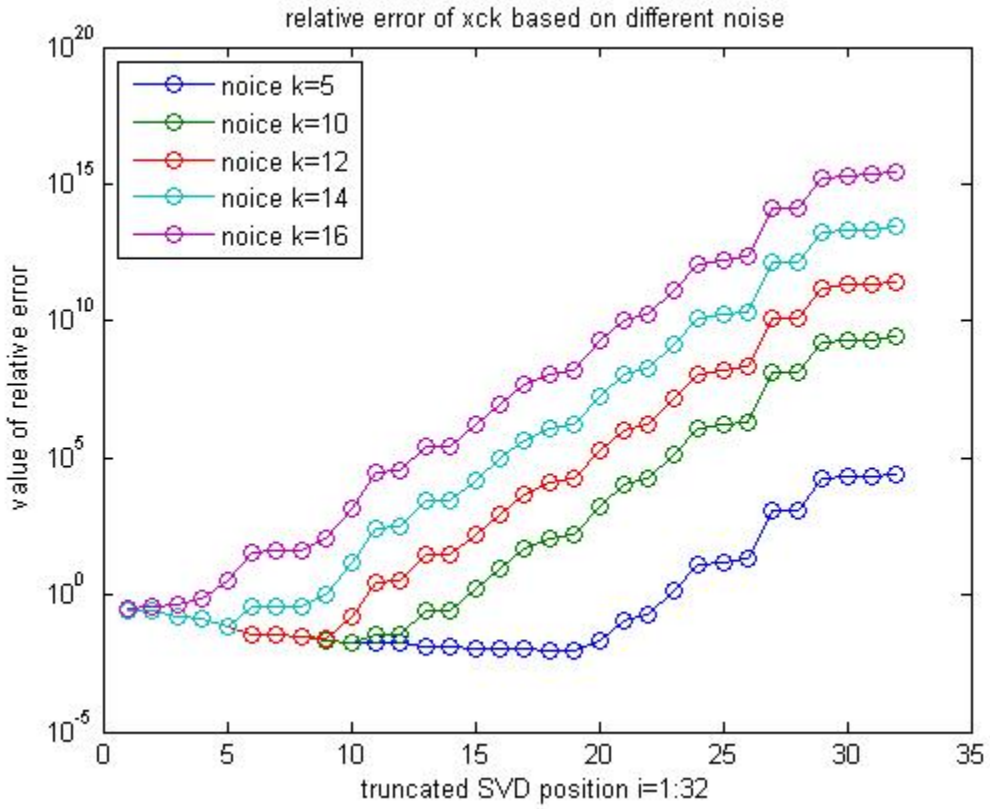
(1) repeat problem 4 by ***bn*** vectors;

k	5	10	12	14	16
noise level	1.23065e-12	1.23065e-07	1.23065e-05	0.00123	0.12306
relative error	5.36527e+05	5.36529e+10	5.36529e+12	5.36529e+14	5.36529e+16

From the large relative errors, we can still easily see that the results computed by ***backslash*** commend is very far away from the true solutions, which implies that directly using ***backslash*** commend is very unstable in solving this ill-conditioned problem.

(2) repeat problem 5

We already showed the singular values of matrix **A** in in graph in problem 5, now on each different level of noise, we compute each relative error of each solution got by each truncated position from 1 to 32, showed in the graph below:



And we list the best truncated SVD position along with its corresponding minimum value of relative error in the table below:

k	5	10	12	14	16
noise level	1.23065e-12	1.23065e-07	1.2307e-05	0.00123	0.12306
Truncated position	18	10	9	5	1
relative error	0.0086	0.0178	0.0253	0.0724	0.2915

Note that every time we run the program (listed in the appendix), the results we may get maybe a little different, that's is because the commend ***randn*** will randomly generate a sequences of numbers under Gaussian Distribution, that's where the differences come from.

(3) repeat problem 6-1

We solve the normal equation by ***backslash*** and get the minimum relative errors for each λ based on different noise are as follows:

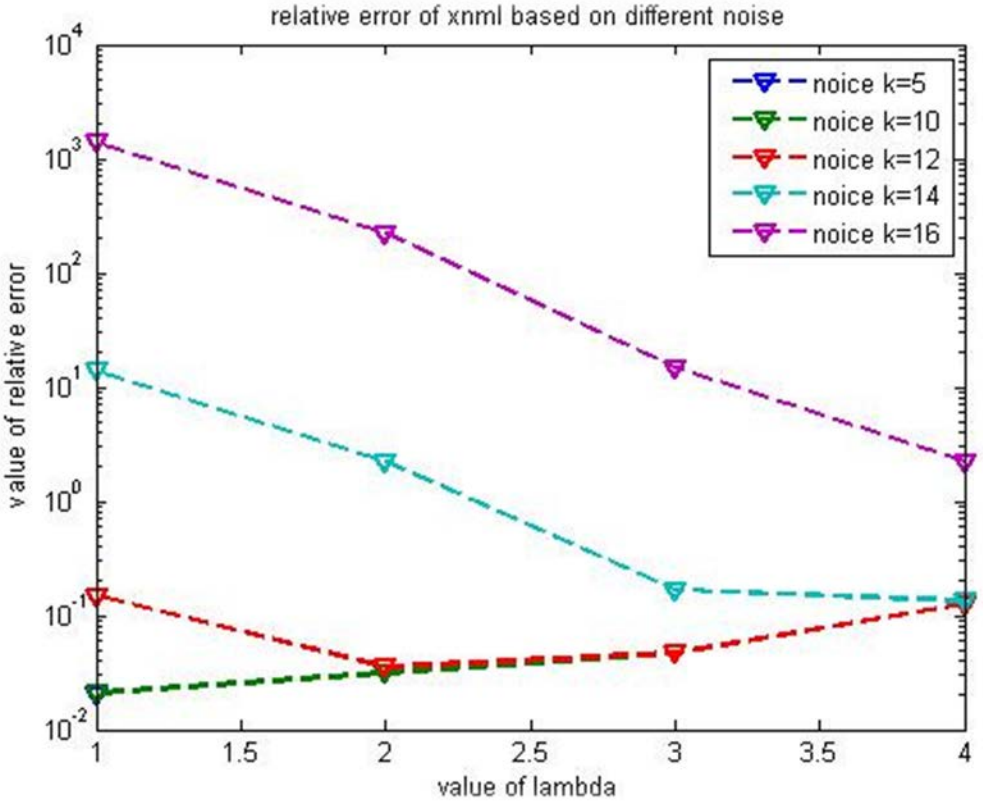
k	5	10	12	14	16
noise level	1.23065e-12	1.23065e-07	1.2307e-05	0.00123	0.12306
λ	0.0001	0.0001	0.001	0.1	0.1
minimum relative error	0.0203	0.0210	0.0356	0.1336	2.2398

For problem 6-2

We solve the normal equation by ***backslash*** and get the minimum relative errors for each λ based on different noise are as follows:

k	5	10	12	14	16
noise level	1.23065e-12	1.23065e-07	1.2307e-05	0.00123	0.12306
λ	0.0001	0.0001	0.001	0.1	0.1
minimum relative error	0.0203	0.0210	0.0356	0.1336	2.2398

As mentioned above in problem5, the two methods are equivalent and thus the results listed in the two tables are exactly the same.


relative error of xnml based on different noise
noice k=5
noice k=10
noice k=12
noice k=14
noice k=16
value of relative error
value of lambda
10^4
10^3
10^2
10^1
10^0
10^-1
10^-2
1
1.5
2
2.5
3
3.5
4

Appendix

Problem: 1.m

```
clc;clear
format compact
format short
b=[];
d=[0.2:0.1:0.8];
for i=1:length(d)
    [A,bb,x]=gravity(32,2,0,1,d(i));
    b=[b,bb];
end
plot(b,'-o')
legend('d=0.2','d=0.3','d=0.4','d=0.5','d=0.6','d=0.7','d=0.8')
ylabel('value of b')
title('relation between d and b')
max_b=max(b);
min_b=min(b);
```

Problem: 2.m

```
clc;clear
format compact
format long
n=[5:5:50]
condA=[];
rankA=[];
for i=1:length(n)
    [A,bb,x]=gravity(n(i),2,0,1,0.5);
    condA=[condA,cond(A)];
    rankA=[rankA,rank(A)];
end
condA;
rankA;
%p=plot(n,condA,'*')
[AX,H1,H2]=plotyy(n,condA,n,rankA,'plot');
xlabel('value of n')
title('relation between n and condition number of A')
set(get(AX(1),'Ylabel'),'String','condition number of A')
set(get(AX(2),'Ylabel'),'String','rank of A')
set(H1,'LineStyle','-','LineWidth',5)
set(H2,'LineStyle','--','LineWidth',5)
legend('condition #','rank','Location','NorthWest')
```

Problem: 3.m

```
clc;clear
format compact
format long
condA=[];
```

```
rankA=[];
d=[0.1:0.1:1];
for i=1:length(d)
    [A,bb,x]=gravity(32,2,0,1,d(i));
    condA=[condA,cond(A)];
    rankA=[rankA,rank(A)];
end
[AX,H1,H2]=plotyy(d,rankA,d,condA,'plot');
xlabel('value of depth d')
title('relation between condition number of A and depth d  ')
set(get(AX(1),'Ylabel'),'String','rank of A')
set(get(AX(2),'Ylabel'),'String','condition number of A')
set(H1,'LineStyle','--','MarkerEdgeColor','b','LineWidth',5)
set(H2,'LineStyle','-','MarkerEdgeColor','r','LineWidth',5)
legend('rank(A)','condition #','Location','NorthEast')
```

Problem: 4.m &5.m

```
clc;clear
format compact
n=32;
d=0.5;
[A,b,x]=gravity(n,2,0,1,d);
% question 4
xc=A\b;
rlerrxc=norm(xc-x)/norm(x)

% question 5
[U,S,V]=svd(A);
subplot(1,2,1); semilogy(diag(S),'*')
title('singular values of A')
legend('n=32')
rlerrxck=[];
for k=1:n
    xck=ktsvd(A,b,n,k);
    new=norm(xck-x)/norm(x);
    rlerrxck=[rlerrxck,new];
end
subplot(1,2,2); semilogy(rlerrxck,'o')
title('relative error of xck')
legend('k=1:32')
minvalue=min(rlerrxck')
k=find(rlerrxck==minvalue)
rlerrxck(16)
```

The ktsvd.m

```
function xck = ktsvd(A,b,n,k)
xck=zeros(n,1);
[U,S,V] = svd(A);
for i=1:k
```

```
   xck=xck+(U(:,i)'*b/S(i,i))*V(:,i);
end
```

Problem: 6.m

```
clc;clear
format compact
lambda=[0.0001,0.001,0.01,0.1];
n=32;
d=0.5;
[A,b,x]=gravity(n,2,0,1,d);

% question 6-1
rlerrxmnl=[];
for i=1:length(lambda)
    xnml=(A'*A+lambda(i)^2.*eye(n))\(A'*b);
    new=norm(xnml-x)/norm(x);
    rlerrxmnl=[rlerrxmnl,new];
end

% question 6-2
rlerrxt=[];
for i=1:4
    c=[A;lambda(i)*eye(n)];
    d=[b;zeros(n,1)];
    xt=lsqlin(c,d);
    new=norm(xt-x)/norm(x);
    rlerrxt=[rlerrxt,new];
end

plot(lambda,rlerrxmnl,'b<',lambda,rlerrxt,'r>')
legend('relative error of x normal equ','relative error of x least
squ','Location','NorthWest')
xlabel('value of lambda')
ylabel('relative error')
```

Problem: 7.m

```
clear;clc
format compact
n=32;
d=0.5;
[A,b,x]=gravity(n,2,0,1,d);
k=[5 10 12 14 16];
s=randn(32,1);
noise=eps*(10.^k);

%repeat question 4
rlerrxcn=[];
noiselevel=[];
for i=1:5
    bn=b+s.*noise(i);
```

```
    new1=norm(noise(i))/norm(b);
    noiselevel=[noiselevel,new1];
    xcn=A\bn;
    new2=norm(xcn-x)/norm(x);
    rlerrxcn=[rlerrxcn,new2];
end

%repeat question 5
[U,S,V]=svd(A);
rlerrxck=[];
positionk=[];
for i=1:5
    bn=b+s.*noise(i);
    rlerr=[];
    for k=1:32
        xckn=ktsvd(A,bn,n,k);
        new=norm(xckn-x)/norm(x);
        rlerr=[rlerr;new];
    end
    rlerrxck=[rlerrxck,rlerr];
    minvalue=min(rlerrxck(:,i));
    locate=find(rlerrxck(:,i)==minvalue);
    positionk=[positionk,locate];
end
min(rlerrxck)
positionk
semilogy(rlerrxck,'-o')
title('relative error of xck based on different noise')
legend('noice k=5','noice k=10','noice k=12','noice k=14','noice
k=16','Location','NorthWest')
xlabel('truncated SVD position i=1:32')
ylabel('value of relative error')

%repeat question 6-1
lambda=[0.0001,0.001,0.01,0.1];
rlerrxmnl=[];
positionlmd=[];
for i=1:5
    bn=b+s.*noise(i);
    rlerr=[];
    for j=1:4
        xnml=(A'*A+lambda(j)^2.*eye(n))\(A'*bn);
        new=norm(xnml-x)/norm(x);
        rlerr=[rlerr;new];
    end
    rlerrxmnl=[rlerrxmnl,rlerr];
    minvalue=min(rlerrxmnl(:,i));
    locate=find(rlerrxmnl(:,i)==minvalue);
    positionlmd=[positionlmd,locate];
end
min(rlerrxmnl)
positionlmd
lambda(positionlmd)

%repeat question 6-2
rlerrxt=[];
```

```
positionlmd=[];
for i=1:5
    bn=b+s.*noise(i);
    rlerr=[];
    for j=1:4
        c=[A;lambda(j)*eye(n)];
        d=[bn;zeros(n,1)];
        xt=lsqlin(c,d);
        new=norm(xt-x)/norm(x);
        rlerr=[rlerr;new];
    end
    rlerrxt=[rlerrxt,rlerr];
    minvalue=min(rlerrxt(:,i));
    locate=find(rlerrxt(:,i)==minvalue);
    positionlmd=[positionlmd,locate];
end
min(rlerrxt)
positionlmd
lambda(positionlmd)

semilogy(rlerrxmnl,'--v','LineWidth',2)
title('relative error of xnml based on different noise')
legend('noice k=5','noice k=10','noice k=12','noice k=14','noice
k=16','Location','NorthEast')
xlabel('value of lambda')
ylabel('value of relative error')
```