a)

```
function sol =cond_Hil_rand(n)

   R=rand(n);

   H=hilb(n);

   sol=[cond(R), cond(H)];

end
```

A=rand(4);

A=0.276922984960890   0.694828622975817   0.438744359656398   0.186872604554379

0.046171390631154   0.317099480060861   0.381558457093008   0.489764395788231

0.097131781235848   0.950222048838355   0.765516788149002   0.445586200710899

0.823457828327293   0.034446080502909   0.795199901137063   0.646313010111265

n	Random matrix	Hilbert matrix
1	1	1
2	9.1	19.3
3	5.2	5.24*10^2
4	20.66	1.55*10^4

We can say that Hilbert matrices are likely to be ill-conditioned matrices as condition number is growing rapidly. However, for a random matrix, the condition number is really random.

b)

A=hilb(4);

[evec,eval]=eig(A'*A)

evec =

0.029193323163918	0.179186290535595	-0.582075699497238	0.792608291163764
-0.328712055759592	-0.741917790630046	0.370502185067095	0.451923120901600
0.791411145832639	0.100228136951032	0.509578634501801	0.322416398581825
-0.514552750000250	0.638282528191121	0.514048272222161	0.252161169688242

norm(A*evec(:,1))/norm(evec(:,1))

ans = **9.670230402259483e-05**

norm(A*evec(:,2))/norm(evec(:,2))

ans = 0.006738273605761

norm(A*evec(:,3))/norm(evec(:,3))

ans = 0.169141220221450

norm(A*evec(:,4))/norm(evec(:,4))

ans = 1.500214280059243

cond(A) = 1.551373873892814e+04 **%norm(A*evec(:,1))/norm(evec(:,1)) = 9.670230402259483e-05**

A=[1,0,1,2;2,2,3,1;4,3,2,3;7,3,4,1];

norm(A*evec(:,1))/norm(evec(:,1))

ans = 1.165044286012182

norm(A*evec(:,2))/norm(evec(:,2))

ans =1.614314212558824

norm(A*evec(:,3))/norm(evec(:,3))

ans =2.602514361419407

norm(A*evec(:,4))/norm(evec(:,4))

ans =**11.236706832224760**

cond(A)=9.64 **%norm(A*evec(:,4))/norm(evec(:,4))=11.236706832224760**

Using eigenvectors of A'*A can be a good approximation of finding condition number for the well-conditioned matrix but not for ill-conditioned matrix.

c)

A=hilb(4);

[Q,R]=qr(A);

cond(Q)*cond(R)/cond(A)=1.000000000000013

Q =

-0.838116354923494   0.522648373955656 -0.153972761515708 -0.026306682088232

-0.419058177461747 -0.441713323920529   0.727753807365351   0.315680185058791

-0.279372118307831 -0.528821386246471 -0.139505522178664 -0.789200462646984

-0.209529088730873 -0.502071666319607 -0.653609205762985   0.526133641764659

R =

-1.193151755273030 -0.670493083938795 -0.474932601123313 -0.369835470902748

```
        0  -0.118533267487887  -0.125655094630809  -0.117541992762881
        0                   0  -0.006221774060129  -0.009566092949394
        0                   0                   0   0.000187904872059
```

A =

```
   1.000000000000000   0.500000000000000   0.333333333333333   0.250000000000000
   0.500000000000000   0.333333333333333   0.250000000000000   0.200000000000000
   0.333333333333333   0.250000000000000   0.200000000000000   0.166666666666667
   0.250000000000000   0.200000000000000   0.166666666666667   0.142857142857143
```

A=hilb(3);

[Q,R]=qr(A);

cond(Q)*cond(R)/cond(A)=0.999999999999999

A=hilb(2);

[Q,R]=qr(A);

cond(Q)*cond(R)/cond(A)= 1

A=hilb(1);

[Q,R]=qr(A);

cond(Q)*cond(R)/cond(A)= 1

We know that stability is not determined by the condition of A. Since in theory, QR factorization guarantees good stability we can assume that the stability ratio should be close to 1 for all cases above. As we can see, even if we use an ill-conditioned matrix, the above results are close to one.

d)

A =

```
   0.276922984960890   0.694828622975817   0.438744359656398   0.186872604554379
   0.046171390631154   0.317099480060861   0.381558457093008   0.489764395788231
   0.097131781235848   0.950222048838355   0.765516788149002   0.445586200710899
   0.823457828327293   0.034446080502909   0.795199901137063   0.646313010111265
```

cond(A)=20.6

[evec,eval]=eig(A'*A)

evec =

0.334676075239744 -0.643619658255031   0.599487644869025   0.338171885021787

0.297553923343812 -0.272165609215882 -0.752249872658138   0.521063981626698

-0.758544601226644   0.054355773206988   0.122060286567955   0.637774900973636

0.473353647197038   0.713249652356659   0.244613939051220   0.455384759209411

norm(A*evec(:,4))/norm(evec(:,4))= 1.945232608679346

b=A*evec(:,4)

b =

0.820616813226491

0.647222632489595

1.219114280226397

1.097898530585030

[Q,R]=qr(A);

y=Q\b;

x=R\y

x =

0.338171885021787

0.521063981626699

0.637774900973635

0.455384759209412

evec(:,4)-x

ans =

1.0e-14 *

-0.077715611723761

-0.077715611723761

0.122124532708767

-0.083266726846887

The error is very small as A is a well-conditioned matrix.

<ill-condition matirx>

A=hilb(4);

b=evec(:,2);

y=Q\b;

x=R\y

x =

   1.0e+02 *

   0.265923144456661

  -1.101050260270333

   0.148744534696545

   0.947249348073386

evec(:,2)-x

ans =

   1.0e+02 *

  -0.264131281551305

   1.093631082364033

  -0.147742253327035

  -0.940866522791474

The error is much larger than the well-conditioned matrix case. We found that solving equations using QR factorization is very stable in question c. So the error should really depend on the condition of the matrix From question a) we know that the first matrix A is well-conditioned so the error should be very small. Second A is ill-conditioned so an error is likely to be large.