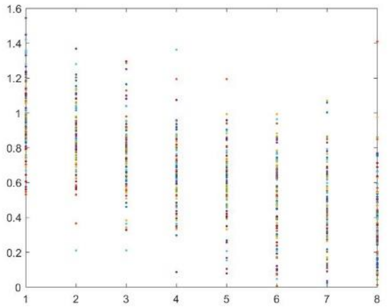
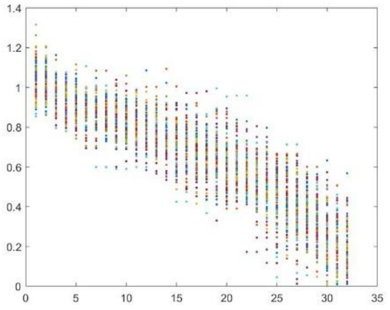
Problem 12.3
(a)

```
num_size_mat = 4; num_rand = 100;
% num_size_mat means the number of different sizes of matrix
% we want to compute, and num_rand means the times of random
% matrices we want to take for one specific size of matrix.
rho_mean = zeros(num_size_mat,1); % mean spectral radius
for power = 3 : (3 + num_size_mat - 1)
    figure;
    rho_vals = zeros(num_rand,1);
    m = 2^power; % size of matrix
    eig_vals = zeros(num_rand,m);
    for i = 1:num_rand
        A = randn(m,m) / sqrt(m);
        lambda = eig(A);
        eig_vals(i,:) = lambda';
        plot([1:m], abs(eig_vals(i,:)), '.');
        rho_vals(i) = max(abs(eig_vals(i,:)));
        hold on;
    end
    rho_mean(power - 2) = mean(rho_vals);
end
figure;
length(rho_mean)
plot(2 .^ [3:(3+ num_size_mat -1)], rho_mean, 'r-')
```

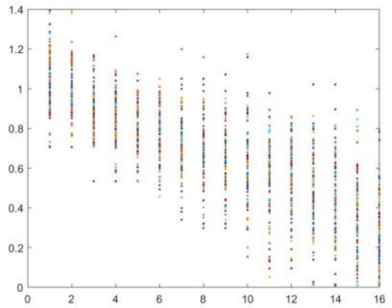
Here is the plot for 100 random matrices' eigenvalues when m=8:



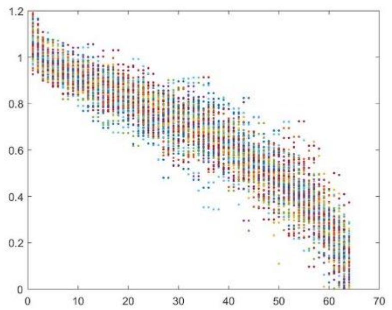
Here is the plot for 100 random matrices' eigenvalues when m=16:



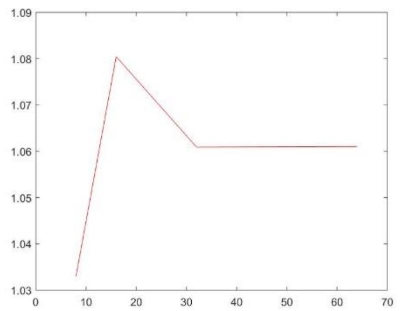
Here is the plot for 100 random matrices’ eigenvalues when m=32:



Here is the plot for 100 random matrices’ eigenvalues when m=64:



The mean spectral radius varies with different m shown as follows:



Notice that there is no pattern of the behavior of spectral radius for a normally distributed random matrix.

(b)

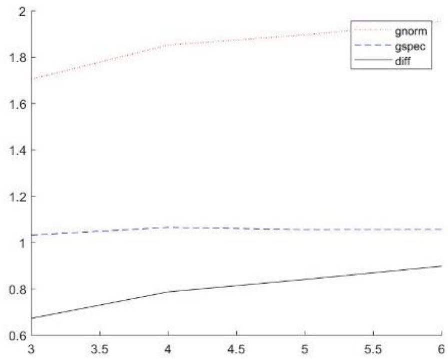
```
j = 1;
clf;
allNorms = [];
allSpecs = [];
idx = 3:6;
for j = idx
    gnorm = [];
    gspec = [];
    for i = 1:100
        m = 2^j;
        A = randn(m,m) / sqrt(m);
        gnorm = [gnorm norm(A)];
        gspec = [gspec max(abs(eig(A)))];
```

```
        hold on;
    end

    allNorms = [allNorms mean(gnorm)];
    allSpecs = [allSpecs mean(gspec)];
end

plot(idx, allNorms, 'r:');
plot(idx, allSpecs, 'b--');
plot(idx, allNorms-allSpecs, 'k-');
legend('gnorm', 'gspec', 'diff');
```

The behavior of 2-norm of a random matrix is as follows:



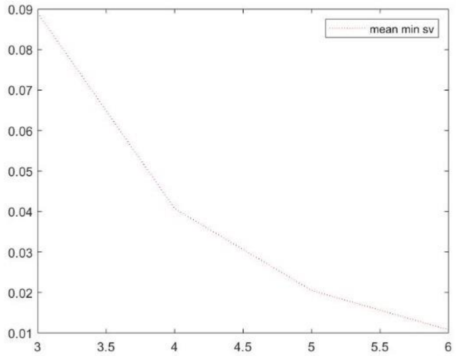
Here "gnorm" represents the mean norm for random matrices with fixed m=8,16,32,64, and "gspec" represents the mean spectral radius for random matrices with fixed m=8,16,32,64. Notice the difference between them doesn't show that they are approaching as m increasing since the difference itself is increasing.

(c)

```
j = 1;
clf;
allSV = [];
idx = 3:6;
for j = idx
    gSV = [];
    for i = 1:100
        m = 2^j;
        A = randn(m,m) / sqrt(m);
        gSV = [gSV min(svd(A))];
        hold on;
    end
    sv_total = length(gSV);
    t = []; y = [];
    for k = idx
        t = [t 2^(9 - k) ^(-1)];
        y = [y sum(sum(gSV < t(k - 2)))];
```

```
    end
    plot(t, y, 'b--');
    legend('proportion of min sv < x')
    allSV = [allSV mean(gSV)];
    figure;
end
plot(idx, allSV, 'r:');
legend('mean min sv');
```

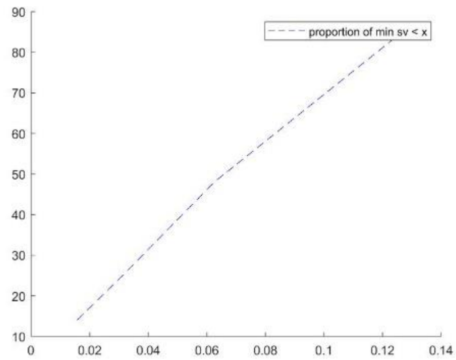
The smallest singular value behaves as follows:



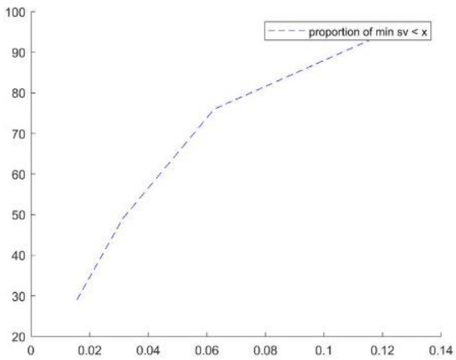
Notice that as m increasing, the smallest singular value will decrease.

For fixed m, here are the plots for proportions of smallest singular values for 100 random matrices:
Here we choose the threshold as 8^{-1}, 16^{-1}, 32^{-1}, 64^{-1}.

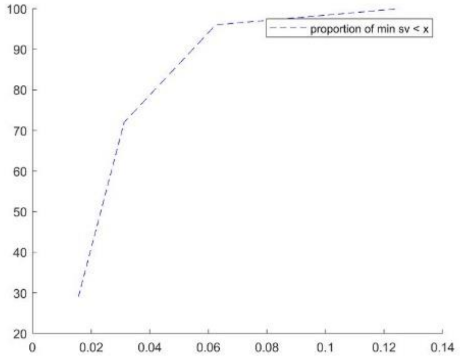
When m = 8:



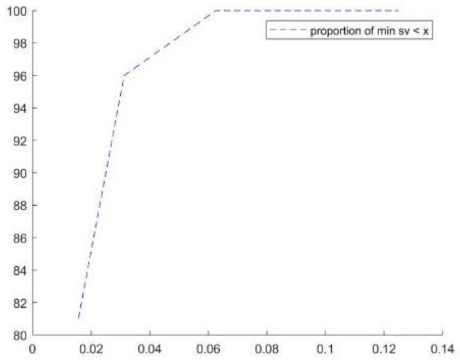
When m=16



When m=32:



When m =64:



Notice that with m increasing, the proportions of random matrices with smaller singular values will get larger.

(d)

If the matrices are upper triangular matrices, the behavior of the curves plotted is even more marked, but the direction should not change.

Lecture 19 Examples with m=50, n=10

```
function [Q,R] = mgs(A)

[m,n] = size(A);
if (m < n)
        print("Error: rows more than cols!")
end

Q = zeros(m,n); R = zeros(n,n);

for i = 1 : n
        Q(:,i) = A(:,i);
end

for i = 1 : n
        R(i,i) = norm(Q(:,i),2);
        Q(:,i) = Q(:,i) / R(i,i);
        for j = (i + 1) : n
                R(i,j) = Q(:,i)' * Q(:,j);
                Q(:,j) = Q(:,j) - R(i,j) * Q(:,i);
        end
```

```
end
end
```

Main Function:

```
m = 50; n = 10;
t = (0:m-1)' / (m-1);
A = [];
for i = 1:n
        A = [A t.^(i-1)];
end
b = exp(sin(4*t));

% Conditioning
x = A \ b; y = A * x;
kappa = cond(A)
theta = asin(norm(b - y) / norm(b))
eta = norm(A) * norm(x) / norm(y)

% Householder
[Q,R] = qr(A,0);
x = R \ (Q' * b);
s1 = x(10)

[Q2,R2] = qr([A b],0);
R2_new = R2(1:n, 1:n);
Qb = R2(1:n, n+1);
x = R2_new \ Qb;
s2 = x(10)

x = A \ b;
s3 = x(10)

% Gram-Schmidt
[Q,R] = mgs(A);
x = R \ (Q' * b);
s4 = x(10)

[Q2,R2] = mgs([A b]);
R2_new = R2(1:n,1:n);
Qb = R2(1:n,n+1);
x = R2_new \ Qb;
s5 = x(10)

% Normal equations
x = (A' * A) \ (A' * b);
s6 = x(10)

% SVD
[U,S,V] = svd(A,0);
x = V * (S \ (U' * b));
s7 = x(10)
```

Answer:

```
kappa =
         3.558944505096530e+06

theta =
         2.371557783639898e-04
```

```
eta =
          5.167276815543769e+03

s1 =
        -5.225852597923072e+02

s2 =
        -5.225852597965024e+02

s3 =
        -5.225852597959299e+02

s4 =
        -5.225852456387123e+02

s5 =
        -5.225852597883538e+02

s6 =
        -5.225311686707603e+02

s7 =
        -5.225852597923067e+02
```