Table of Contents

ECEN 410 - Assignment 3

David Dobbie

```
clc
clear

set(0,'defaultTextInterpreter','latex');
```

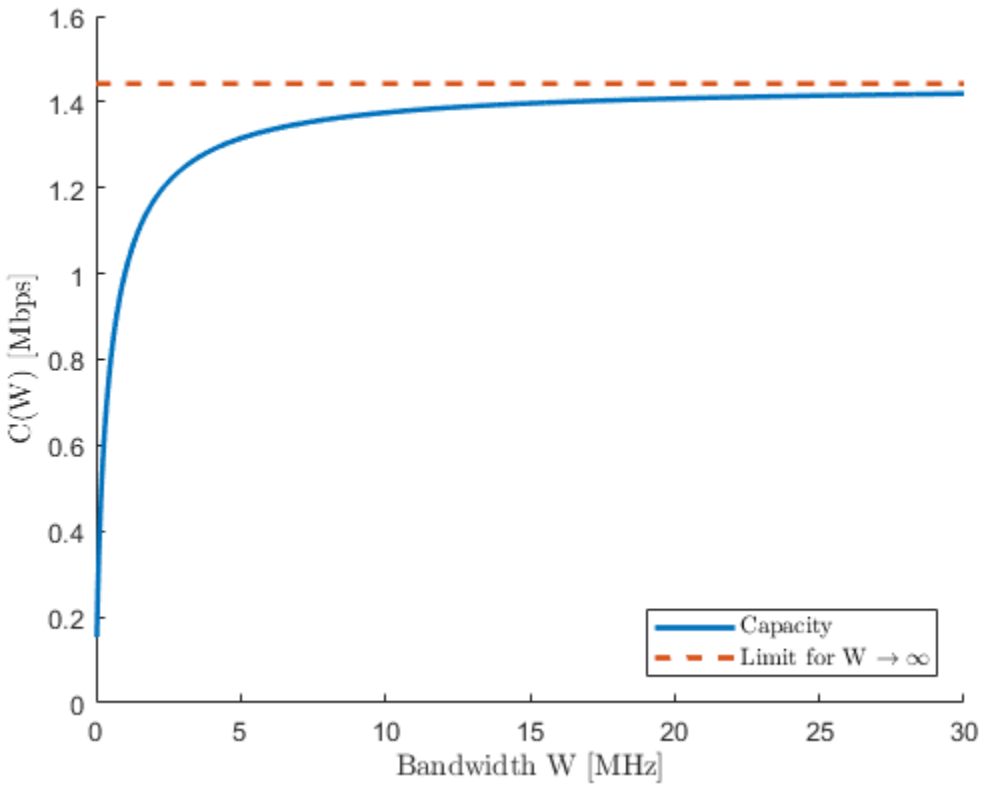
Question 1

Slide 18

Set P/No to 1x10^6

```
length = 1000;

SNR_0 = 1;
W = linspace(0,30,length);
C = W .* log2(1 + SNR_0./W);

asymptotic_C = SNR_0 * log2(exp(1));
asymptotic_C = asymptotic_C*ones(1,length);

figure(1)
clf
hold on
p1=plot(W,C, '-');
p1.LineWidth = 2;
p2 = plot(W,asymptotic_C, '--');
p2.LineWidth = 2;
hold off
xlabel('Bandwidth W [MHz]')
ylabel('C(W) [Mbps]')
xlim([0 30])
ylim([0 1.6])
lgnd = legend('Capacity', 'Limit for W $\rightarrow\infty
$','Location','SouthEast');
```

```
set(lgnd, 'Interpreter', 'latex')
```

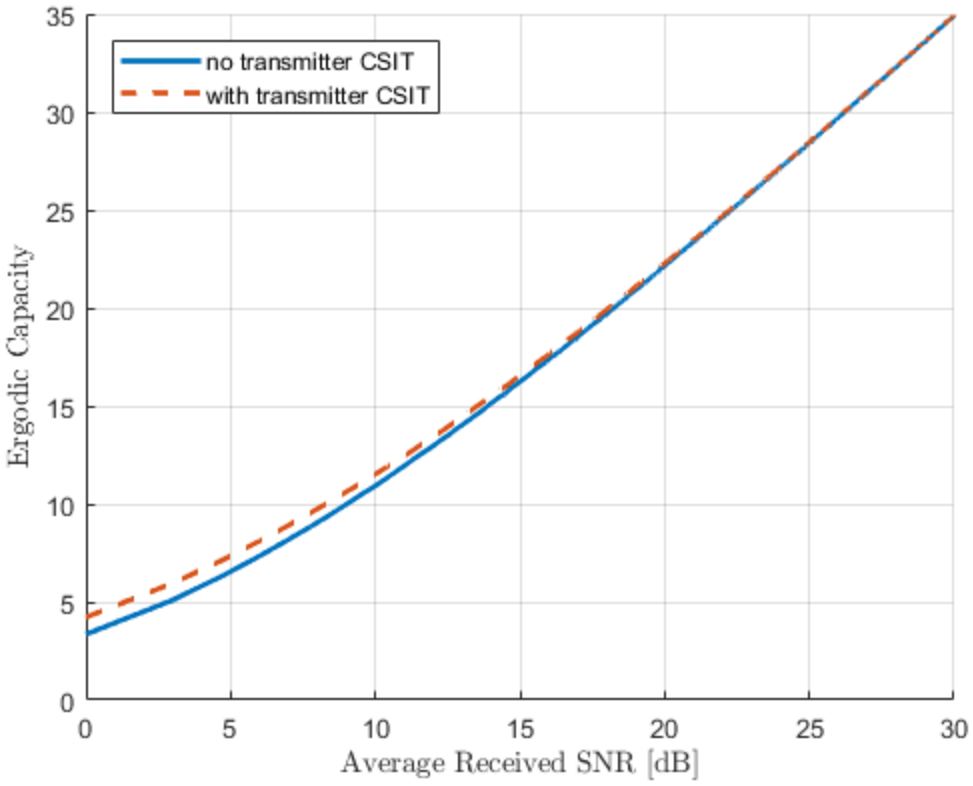


Slide 45

Plot the Ergodic MIMO capacity for a 4 x 4 system

```
clc
clear


% no CSIT (equal powers)
length = 1000;
H_trials = 1e4;

avg_SNR_lin = linspace(db2pow(0), db2pow(30), length);

ergodic_cap_equal_filling = zeros(1,length);

for SNR_indx = 1:length
    Rx = avg_SNR_lin(SNR_indx)*eye(4,4); %optimum Rx is diagonal
    capacity = zeros(1,H_trials);
    for cap_idx = 1:H_trials
        % create random unit var, zero mean, normally distirbuted
 channel
        % normalise across 8 dimensional Gaussian to unit variance
        H = sqrt(1/(2*4))*(randn(4) + 1j*randn(4));
```

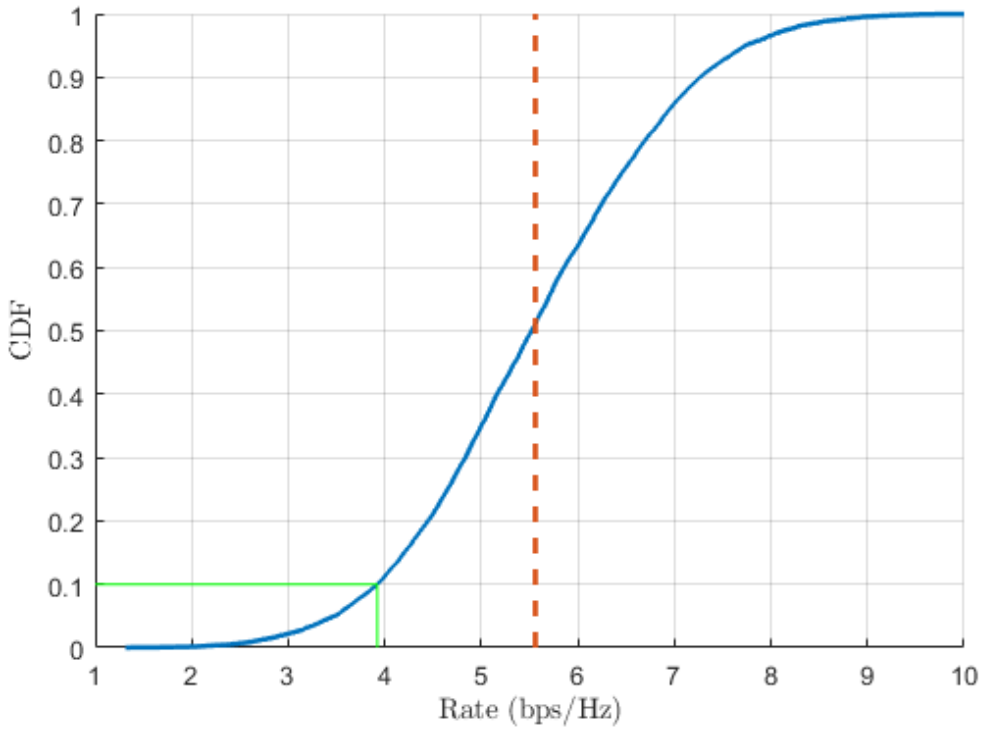
```
        term = eye(4) + H*Rx*ctranspose(H);
        % add absolute term to make the logarithm behave
        capacity(cap_idx) = log2(abs(det(term)));
    end
    ergodic_cap_equal_filling(SNR_indx) = mean(capacity);
end

% CSIT (inequal powers)
ergodic_cap_inequal_filling =  waterFillingResult(avg_SNR_lin,
 H_trials);


figure(2)
clf
hold on
p1 = plot(pow2db(avg_SNR_lin), ergodic_cap_equal_filling, '-');
p1.LineWidth = 2;
p2 = plot(pow2db(avg_SNR_lin), ergodic_cap_inequal_filling, '--');
p2.LineWidth = 2;
hold off
xlabel('Average Received SNR [dB]')
ylabel('Ergodic Capacity')
lgnd = legend('no transmitter CSIT', 'with transmitter CSIT',...
'Location', 'NorthWest');
grid on
ylim([0 35])
```



slide 46

```
SNR_dB = 10;

Rx = db2pow(SNR_dB)*eye(2);

H_trials = 1e4;
capacity = zeros(1,H_trials);
for cap_idx = 1:H_trials
    % create random unit var, zero mean, normally distirbuted channel
    % normalise across 4 dimensional Gaussian to unit variance
    H = sqrt(1/(2*2))*(randn(2) + 1j*randn(2));
    term = eye(2) + H*Rx*ctranspose(H);
    % add absolute term to make the logarithm behave
    capacity(cap_idx) = log2(abs(det(term)));
end
ergodic_cap_mean = mean(capacity);

figure(3)
clf
hold on
p1 = cdfplot(capacity)
p1.LineWidth = 2;

xlim([1 10])

[c index] = min(abs(p1.YData-0.1))
outage = p1.XData; % 10% outage cap
outage_cap = outage(index);

p2a = plot([outage_cap outage_cap],[0 0.1], 'g-');
p2b = plot([0 outage_cap],[0.1 0.1], 'g-');


p3 = plot([ergodic_cap_mean ergodic_cap_mean], [0 1], '--');
p3.LineWidth = 2;

ylabel('CDF')
xlabel('Rate (bps/Hz)')
title('')

grid on
hold off


p1 =

  Line with properties:

            Color: [0 0.4470 0.7410]
        LineStyle: '-'
        LineWidth: 0.5000
           Marker: 'none'
```

```
        MarkerSize: 6
    MarkerFaceColor: 'none'
              XData: [1×20002 double]
              YData: [1×20002 double]
              ZData: [1×0 double]

  Use GET to show all properties


c =

     0


index =

        2001
```

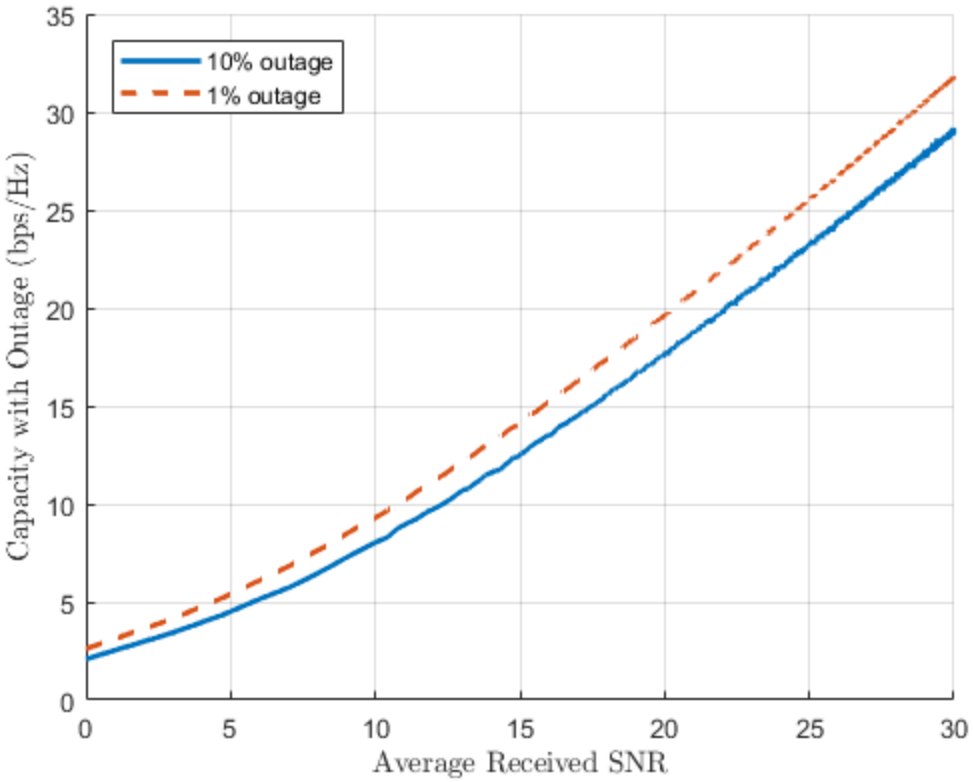


slide 47a

```
clc
clear


length = 1000;
H_trials = 1e4;
```

```
avg_SNR_lin = linspace(db2pow(0), db2pow(30), length);

outage_10_percent = zeros(1,length);
outage_1_percent = zeros(1,length);

for SNR_indx = 1:length
    Rx = avg_SNR_lin(SNR_indx)*eye(4,4); %optimum Rx is diagonal
    capacity = zeros(1,H_trials);
    for cap_idx = 1:H_trials
        % create random unit var, zero mean, normally distirbuted
 channel
        % normalise across 8 dimensional Gaussian to unit variance
        H = sqrt(1/(2*4))*(randn(4) + 1j*randn(4));
        term = eye(4) + H*Rx*ctranspose(H);
        % add absolute term to make the logarithm behave
        capacity(cap_idx) = log2(abs(det(term)));
    end

    [f x] = ecdf(capacity);

    [c, index_10_percent] = min(abs(f-0.1));
    outage_10_percent(SNR_indx) = x(index_10_percent);

    [c, index_1_percent] = min(abs(f-0.01));
    outage_1_percent(SNR_indx) = x(index_1_percent);


end
```

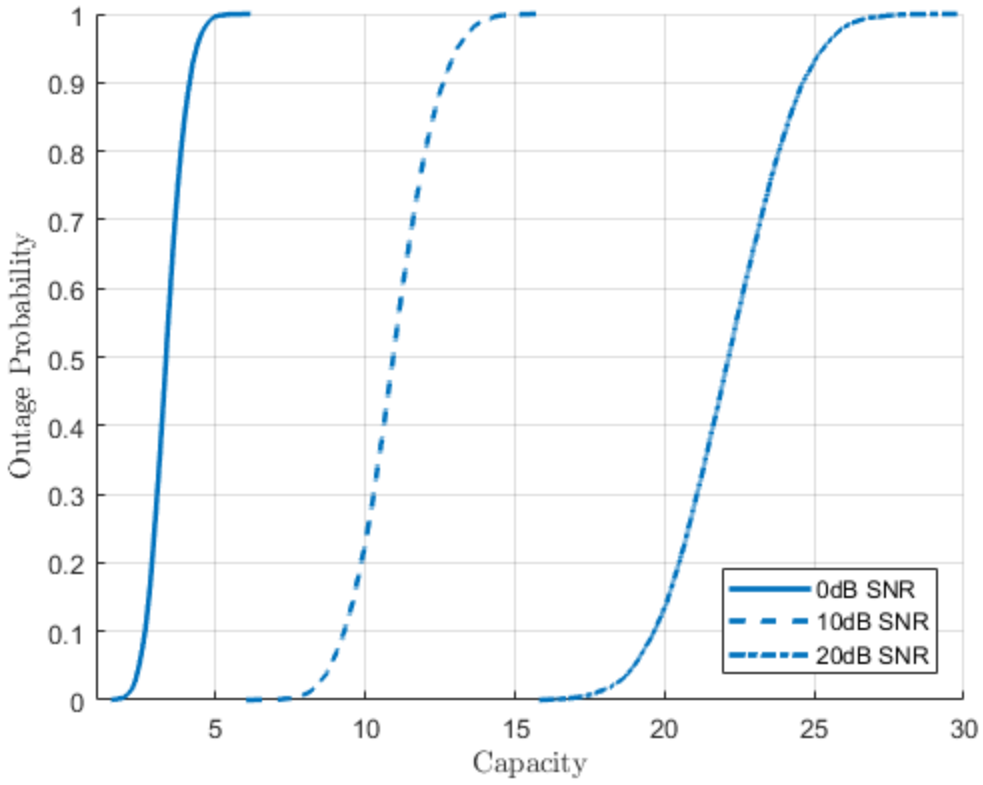
```
figure(4)
clf
hold on
p2 = plot(pow2db(avg_SNR_lin), outage_1_percent);
p2.LineWidth = 2;
p1 = plot(pow2db(avg_SNR_lin), outage_10_percent, '--');
p1.LineWidth = 2;
hold off
grid on
legend('10% outage','1% outage','Location','NorthWest')
xlabel('Average Received SNR')
ylabel('Capacity with Outage (bps/Hz)')
```



slide 47b

```
clc
clear


H_trials = 1e4;
capacity = zeros(3,H_trials);

SNR_dB = [0 10 20];


for SNR_indx = 1:length(SNR_dB)
    SNR_lin = db2pow(SNR_dB(SNR_indx));

    Rx = SNR_lin*eye(4);
    for cap_idx = 1:H_trials
        % create random unit var, zero mean, normally distirbuted
 channel
        % normalise across 4 dimensional Gaussian to unit variance
        H = sqrt(1/(2*4))*(randn(4) + 1j*randn(4));
        term = eye(4) + H*Rx*ctranspose(H);
        % add absolute term to make the logarithm behave
        capacity(SNR_indx, cap_idx) = log2(abs(det(term)));
    end
```

```
end

figure(5)
clf
hold on
p1 = cdfplot(capacity(1,:));
p1.LineWidth = 2;
p1.Color = [0 0.4470 0.7410];

p2 = cdfplot(capacity(2,:));
p2.LineWidth = 2;
p2.Color = [0 0.4470 0.7410];
p2.LineStyle = '--';

p3 = cdfplot(capacity(3,:));
p3.LineWidth = 2;
p3.Color = [0 0.4470 0.7410];
p3.LineStyle = '-.';


xlim([1 30])



ylabel('Outage Probability')
xlabel('Capacity')
title('')

legend('0dB SNR', '10dB SNR', '20dB SNR','Location', 'SouthEast')

grid on
hold off
```

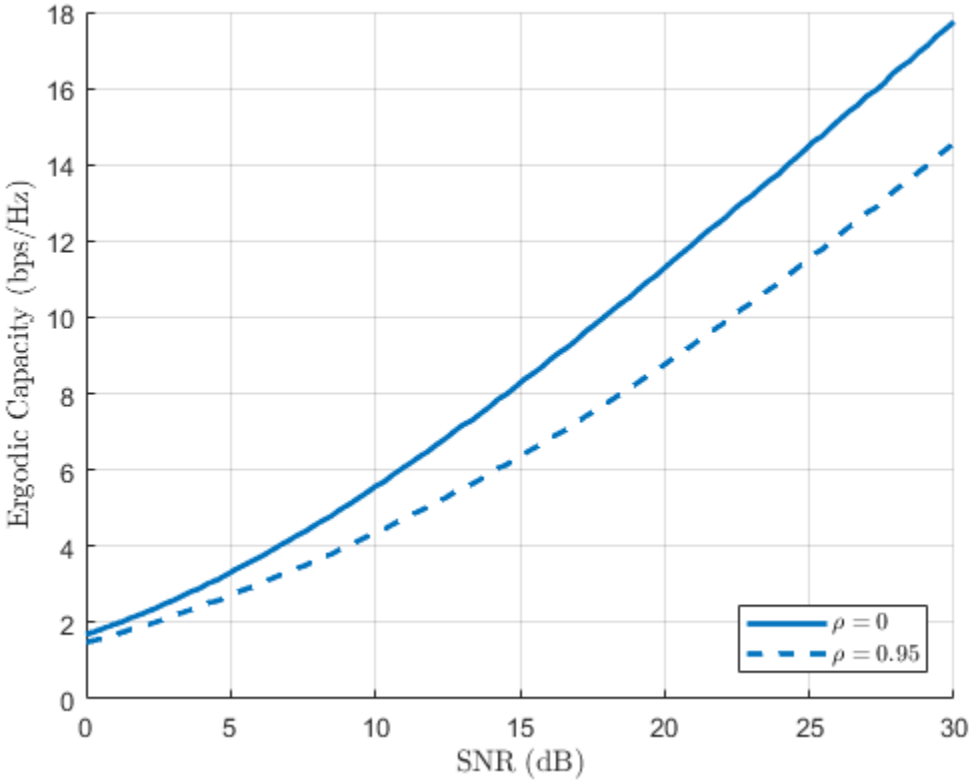


slide 50

```
clc
clear

% no CSIT (equal powers)
length = 100;
H_trials = 1e4;

avg_SNR_lin = logspace(0, log10(db2pow(30)), length);

ergodic_cap_no_correlation = zeros(1,length);
ergodic_cap_correlation = zeros(1,length);


for SNR_indx = 1:length
    Rx_nocorr = avg_SNR_lin(SNR_indx)*eye(2,2); %optimum Rx is
 diagonal
    Rx_corr = avg_SNR_lin(SNR_indx)*[1 0.95; 0.95 1]; %Rx with
 corrleation, non_diag
    capacity = zeros(2,H_trials);

    for cap_idx = 1:H_trials
        % create random unit var, zero mean, normally distirbuted
 channel
```

```
        % normalise across 8 dimensional Gaussian to unit variance
        H = sqrt(1/(2*2))*(randn(2) + 1j*randn(2));


        % have no correlation
        term = eye(2) + H*Rx_nocorr*ctranspose(H);
        % add absolute term to make the logarithm behave
        capacity(1, cap_idx) = log2(abs(det(term)));

        %have correlation
        term = eye(2) + H*Rx_corr*ctranspose(H);
        % add absolute term to make the logarithm behave
        capacity(2, cap_idx) = log2(abs(det(term)));
    end

    ergodic_cap_no_correlation(SNR_indx) = mean(capacity(1,:));
    ergodic_cap_correlation(SNR_indx) = mean(capacity(2,:));
end


figure(6)
clf
hold on
p1 = plot(pow2db(avg_SNR_lin), ergodic_cap_no_correlation);
p1.LineWidth = 2;
p1.Color = [0 0.4470 0.7410];

p2 = plot(pow2db(avg_SNR_lin), ergodic_cap_correlation);
p2.LineWidth = 2;
p2.Color = [0 0.4470 0.7410];
p2.LineStyle = '--';

xlim([0 30])


ylabel('Ergodic Capacity (bps/Hz)')
xlabel('SNR (dB)')
title('')

lgnd = legend('$\rho = 0$', '$\rho = 0.95$','Location', 'SouthEast')
set(lgnd, 'Interpreter', 'latex')
grid on
hold off


lgnd =

  Legend ($\rho = 0$, $\rho = 0.95$) with properties:

        String: {'$\rho = 0$'  '$\rho = 0.95$'}
      Location: 'southeast'
    Orientation: 'vertical'
      FontSize: 9
      Position: [0.6869 0.1413 0.1946 0.0869]
```

```
        Units: 'normalized'

  Use GET to show all properties
```



functions

```
function cap_result = waterFillingResult(SNR_avg_lin, H_trials)

    capacity = zeros(1,H_trials);
    for SNR_indx = 1:length(SNR_avg_lin)

        SNR_budget = SNR_avg_lin(SNR_indx);


        for cap_idx = 1:H_trials
            % create random unit var, zero mean, normally distirbuted
 channel
            % normalise across 8 dimensional Gaussian to unit variance
            H = sqrt(1/(2*4))*(randn(4) + 1j*randn(4));

            eigenval = eig(H*H');
            % we set noise variance to 1
            % sets max possible power on each antenna
            P = waterfilling_process(eigenval, SNR_budget*4);
```

```
            capacity(cap_idx) = log2 ( prod ( 1 + P.*eigenval));
        end
        cap_result(SNR_indx) = mean(capacity);
    end
end


function P = waterfilling_process(eigen_val, SNR_budget)
    %assume noise variance is unity throughout process
    % therefore: noise floor is from eigenvals, SNR = P

    M = length(eigen_val);

    P = -ones(M,1);

    mask = ones(4,1); %use this mask to remove channels

    while min(P) < 0 %while any of P has a negative value
        noise_floor = mask.*(1./(eigen_val));
        mu = (SNR_budget + sum(noise_floor)  )/M;
        P = mask.*(mu - noise_floor);
        mask(P<0) = 0; % disregard any negative channels
        M = nnz(mask);
    end
    sum(P);
end
```

Published with MATLAB® R2017b