Table of Contents

Homework 2

```
%{
MECH7710
Matt Boler
%}
```

Problem 1

```
%{
Two random variables x 1 and x 2 have a joint PDF that is uniform
 inside the circle (in
the x 1 -x 2 plane) with radius 2, and zero outside the circle.
a) Find the math expression of the joint PDF function.
b) Find the conditional PDF P x 2 | x 1 ( x 2 | x 1 # 0 . 5 ) ?
c) Are the two random variables uncorrelated?
d) Are the two random variables statistically independent?
%}

% --- Done on paper ---
```

Problem 2

```
%{
The stationary process x(t) has an autocorrelation function of the
 form:
R_x(\tau) = \sigma^2 exp(-\beta |\tau|)
Another process y(t) is related to x(t) by the deterministic equation:
y(t) = ax(t) + b
where the constants a and b are known.
a) What it the autocorrelation function for y(t) ?
b) What is the crosscorrelation function R_xy(\tau)=E[x(t)y(t+\tau)]?
%}

% --- Done on paper ---
```

Problem 3

```
%{
Use least squares to identify a gyroscopes scale factor (a) and bias
 (b). Simulate the
```

```
gyroscope using:
g(k)=ar(k) + b + n(k)
n ~ N(0, \sigma= 0.3 deg/s)
r(k) = 100sin(\omega * t)
a) perform the least squares with 10 samples (make sure to pick # so
 that you get one
full cycle in 10 samples.
b) Repeat part a 1000 times and calculate the mean and standard
 deviation of the
estimate errors (this is known as a Monte Carlo Simulation). Compare
 the results
to the theoretically expected mean and standard deviation
c) Repeat part (a) and (b) using 1000 samples. What does the
 theoretical and Monte
Carlo standard deviation of the estimated errors approach?
d) Set up the problem to run as a recursive least squares and plot the
 coefficients and
theoretical standard deviation of the estimate error and the actual
 estimate error as
a function of time.
%}

a = 0.85; % scale factor
b = 5; % bias (deg/s)
sigma = 0.3; % deg/s

% Lets be really lazy and use a dt of 1 so sample = time
% Just have to use a slow omega
% a. sim with 10 samples
% b. do (a) 1000x

% Parameters chosen:
dt = 1;
w = 360/9; % 1 cycle in 10 samples
t = 0:dt:9;

% Data generation:

angular_velocity = 100 * sind(w * t);

g = zeros(size(t));
g_MC_10 = zeros(length(t), 1000);
n = zeros(size(t));

Y = zeros(size(t'));
H = zeros(size([t', t']));
X_est_10 = zeros(2,1);
X_est_MC_10 = zeros(2, 1000);


for i = 1:1000
    n = randn(size(t)) * sigma;
    g = a * angular_velocity + b + n;
```

```
    Y = g';
    H = [angular_velocity', ones(size(angular_velocity'))];

    %{
    Doesn't matter which run we use as our single and this is a
 convenient
    storage variable
    %}
    X_est_10 = H \ Y;

    % Store history
    X_est_MC_10(:,i) = X_est_10;
    g_MC_10(:,i) = g';
end

figure(1)
plot(t, angular_velocity, t, g);
legend("True : r(k)", "Measured : g(k)");
title("True vs Measured Omega (Single Run)");
xlabel('Time (s)');
ylabel('Angular Velocity (deg/s)');

disp("True Values")
disp(["Scale factor: ", string(a)]);
disp(["Bias: ", string(b)]);

disp("Single Sim Result (10 Samples)")
disp(["Estimated scale factor: ", string(X_est_10(1))]);
disp(["Estimated bias: ", string(X_est_10(2))]);


a_MC_10 = mean(X_est_MC_10(1,:));
b_MC_10 = mean(X_est_MC_10(2,:));
predicted_MC_10 = a_MC_10 * angular_velocity + b_MC_10;
mean_err_MC_10 = mean(g - predicted_MC_10);
std_err_MC_10 = std(g - predicted_MC_10);

disp("MC Sim Results (10 Samples)")
disp(["Estimated scale factor: ", string(a_MC_10)]);
disp(["Estimated bias: ", string(b_MC_10)]);
disp(["Mean error: ", string(mean_err_MC_10)]);
disp(["STD error: ", string(std_err_MC_10)]);

% c. (a) and (b) with 1000 samples

% Parameters chosen:
dt = 1;
w = 360/9; % 1 cycle in 10 samples
t = 0:dt:999;

% Data generation:

angular_velocity = 100 * sind(w * t);
```

```
g = zeros(size(t));
g_MC_1000 = zeros(length(t), 1000);
n = zeros(size(t));

Y = zeros(size(t'));
H = zeros(size([t', t']));
X_est_1000 = zeros(2,1);
X_est_MC_1000 = zeros(2, 1000);


for i = 1:1000
    n = randn(size(t)) * sigma;
    g = a * angular_velocity + b + n;

    Y = g';
    H = [angular_velocity', ones(size(angular_velocity'))];

    %{
    Doesn't matter which run we use as our single and this is a
 convenient
    storage variable
    %}
    X_est_1000 = H \ Y;

    % Store history
    X_est_MC_1000(:,i) = X_est_1000;
    g_MC_1000(:,i) = g';
end

figure(2)
plot(t, angular_velocity, t, g);
legend("True : r(k)", "Measured : g(k)");
title("True vs Measured Omega (Single Run)");
xlabel('Time (s)');
ylabel('Angular Velocity (deg/s)');

disp("True Values")
disp(["Scale factor: ", string(a)]);
disp(["Bias: ", string(b)]);

disp("Single Sim Result (1000 Samples)")
disp(["Estimated scale factor: ", string(X_est_1000(1))]);
disp(["Estimated bias: ", string(X_est_1000(2))]);

a_MC_1000 = mean(X_est_MC_1000(1,:));
b_MC_1000 = mean(X_est_MC_1000(2,:));
predicted_MC_1000 = a_MC_1000 * angular_velocity + b_MC_1000;
mean_err_MC_1000 = mean(g - predicted_MC_1000);
std_err_MC_1000 = std(g - predicted_MC_1000);

disp("MC Sim Results (1000 Samples)")
disp(["Estimated scale factor: ", string(a_MC_1000)]);
disp(["Estimated bias: ", string(b_MC_1000)]);
disp(["Mean error: ", string(mean_err_MC_1000)]);
```

```
disp(["STD error: ", string(std_err_MC_1000)]);

% d. Set up as recursive least squares and plot coefficients, actual
% estimate error, and theoretical estimate error.

X = [1; 0];
P = [1, 0; 0, 10];
R = sigma^2;

recursive_X = X;
recursive_P = [P(1,1); P(2,2)];
errors = zeros(size(t));

for i = 1:length(g)
    y = g(i);
    r = angular_velocity(i);
    H = [r, 1];
    prediction = H * X;

    error = y - prediction;

    % Solve for K
    K = P*H' / (H*P*H' + R);
    X = X + K * error;
    P = inv(inv(P) + H'*inv(R)*H);

    recursive_X(:,i) = X;
    recursive_P(:,i) = [P(1,1); P(2,2)];
    errors(i) = error;
end

state_variance = recursive_P(1,:) + recursive_P(2,:);
estimate_variance = state_variance + R;
estimate_std = sqrt(estimate_variance);

figure(3)
plot(t, recursive_X(1,:), t, recursive_X(2,:), t, errors, t,
 estimate_std, 'r-', t, -estimate_std, 'r-');
title("Recursive Least Squares")
legend("Scale factor", "Bias", "Estimate error", "Estimate std");
```

True Values

```
    "Scale factor: "    "0.85"

    "Bias: "    "5"
```

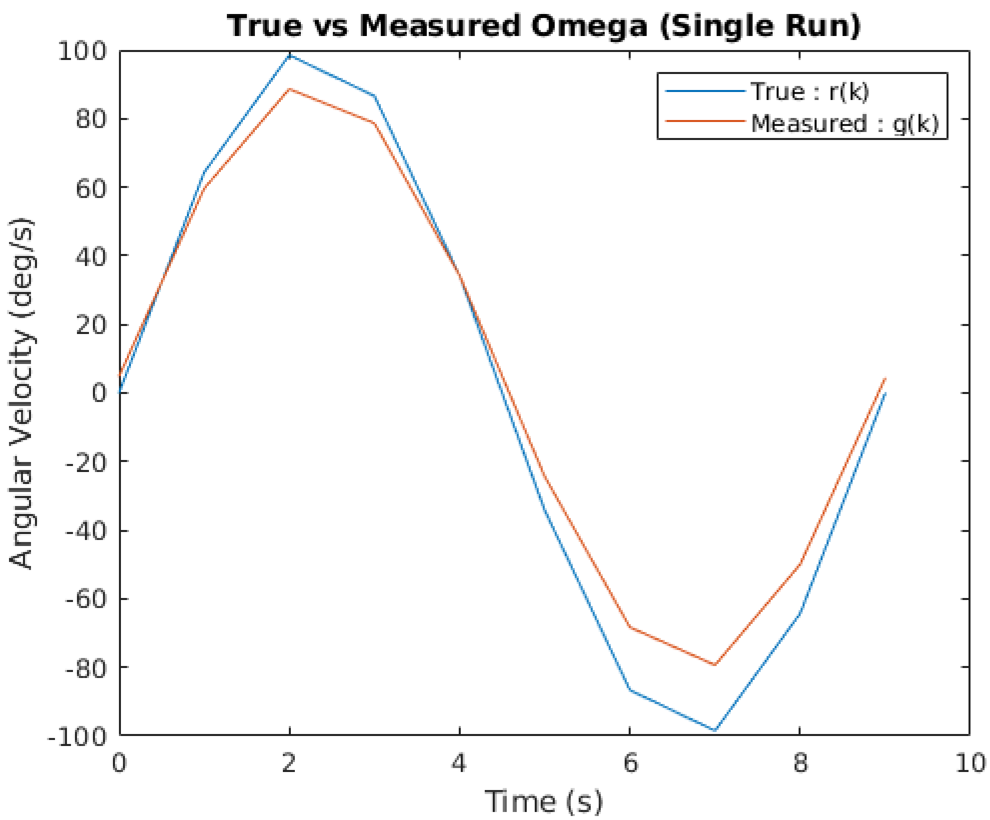
Single Sim Result (10 Samples)

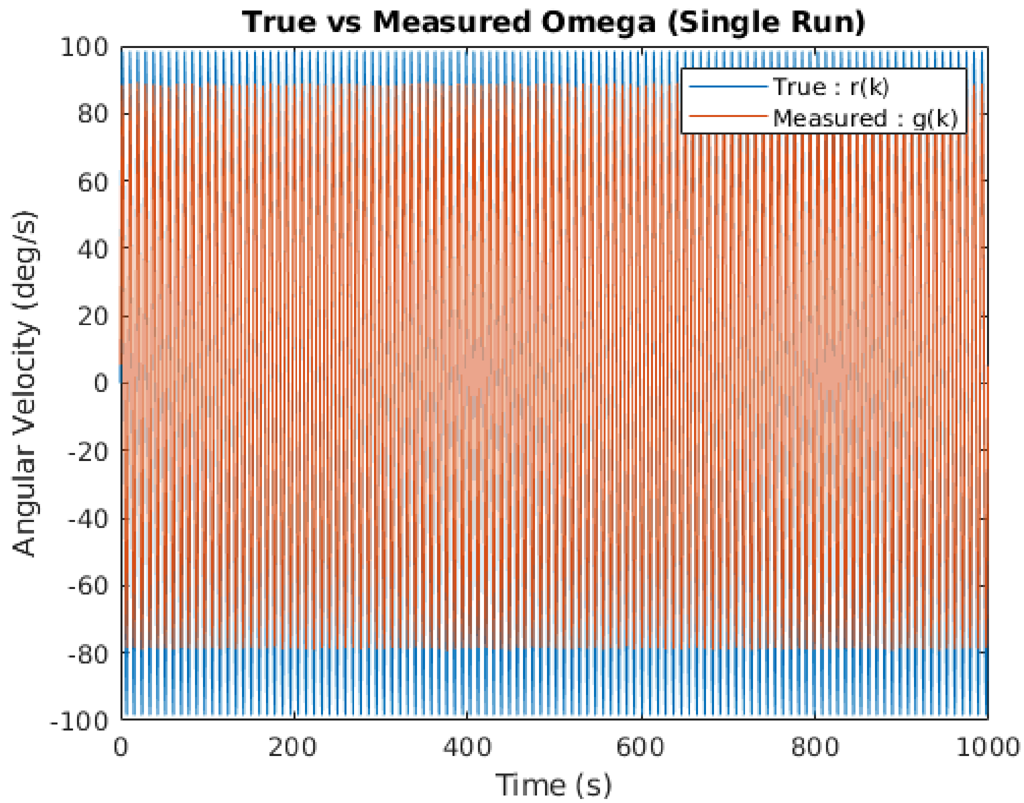
```
    "Estimated scale factor: "    "0.85137"

    "Estimated bias: "    "4.8395"
```

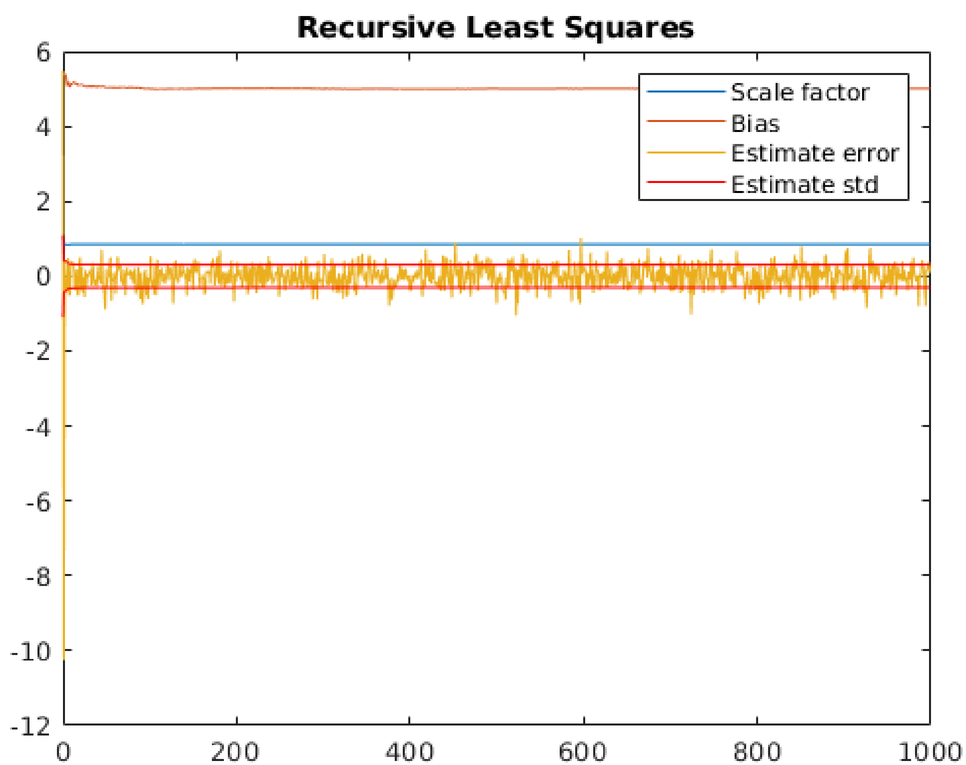
MC Sim Results (10 Samples)

```
    "Estimated scale factor: "    "0.85001"
```

```
    "Estimated bias: "    "5.0015"

    "Mean error: "    "-0.16197"

    "STD error: "    "0.30556"

True Values
    "Scale factor: "    "0.85"

    "Bias: "    "5"

Single Sim Result (1000 Samples)
    "Estimated scale factor: "    "0.85018"

    "Estimated bias: "    "5.0094"

MC Sim Results (1000 Samples)
    "Estimated scale factor: "    "0.85"

    "Estimated bias: "    "5"

    "Mean error: "    "0.0094602"

    "STD error: "    "0.2978"
```




True vs Measured Omega (Single Run)
True : r(k)
Measured : g(k)
Angular Velocity (deg/s)
Time (s)
100
80
60
40
20
0
-20
-40
-60
-80
-100
0
200
400
600
800
1000


Recursive Least Squares
Scale factor
Bias
Estimate error
Estimate std
6
4
2
0
-2
-4
-6
-8
-10
-12
0
200
400
600
800
1000

Problem 4

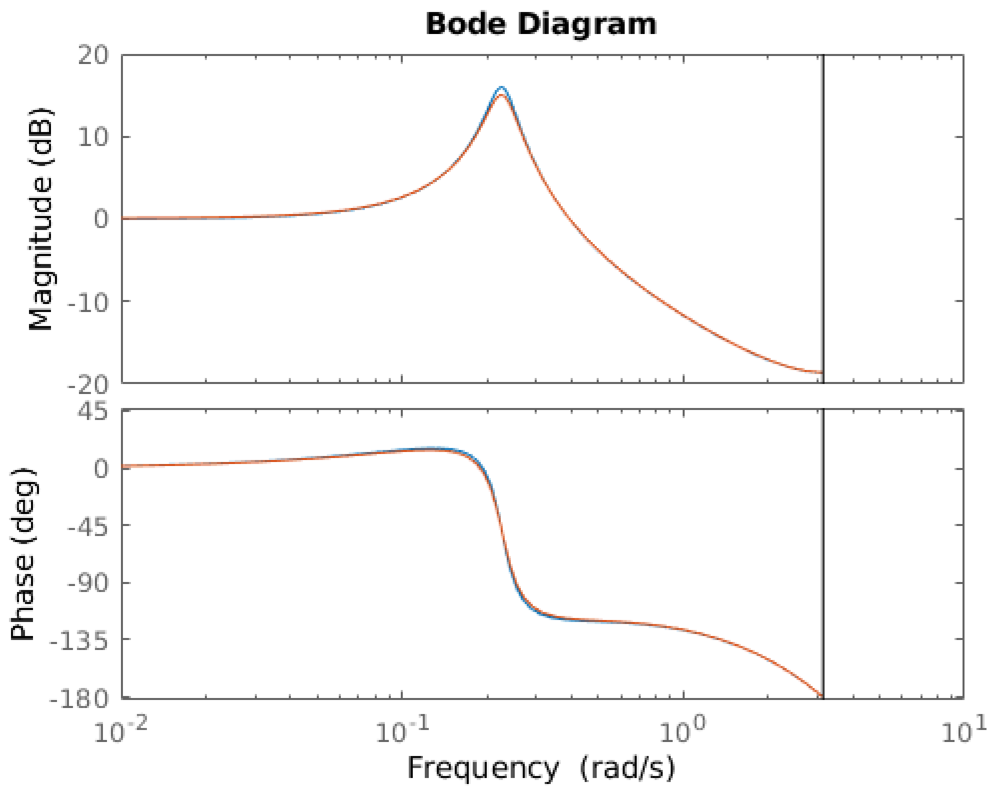
```
%{
Simulate the following discrete system with a
normal random input and output noise:

G(z) = (0.25 * (z-0.8)) / (z^2 - 1.9*z + 0.95)

a) Develop the H matrix for the least squares solution.
b) Use least squares to estimate the coefficients of the above
 Transfer Function.
How good is the fit? Plot the bode response of the I.D. TF and the
 simulated TF
on the same plot. How much relative noise has been added (SNR – signal
 to
noise ratio), plot y and Y on the same plot.
c) Repeat the estimation process about 10 times using new values for
 the noise
vector each time. Compute the mean and standard deviation of your
 parameter
estimates. Compare the computed values of the parameter statistics
 with those
predicted by the theory based on the known value of the noise
 statistics.
d) Now use sigma between 0.1 and 1.0 and repeat parts b and c.
e) What can you conclude about using least squares for sys id with
 large amounts of
noise?
%}
clear all;

numd = 0.25*[1 -0.8];
dend = [1 -1.9 0.95];
true_tf = tf(numd, dend, 1);
u = randn(1000,1);
y = dlsim(numd,dend,u);
sigma = 0.01;
Y = y + sigma * randn(1000,1);


% a.
% From difference equation:
% y(n) = a * y(n-1) + b * y(n-2) + c * u(n-1) + d * u(n-2)
Y_ls = Y(3:end);
H = [Y(2:end-1), Y(1:end-2), u(2:end-1), u(1:end-2)];

%b.
est_coeff_b = H \ Y_ls;
est_n = est_coeff_b(3:4)';
est_d = [1, -est_coeff_b(1:2)'];
est_tf = tf(est_n, est_d, 1);
est_y = dlsim(est_n, est_d, u);
```

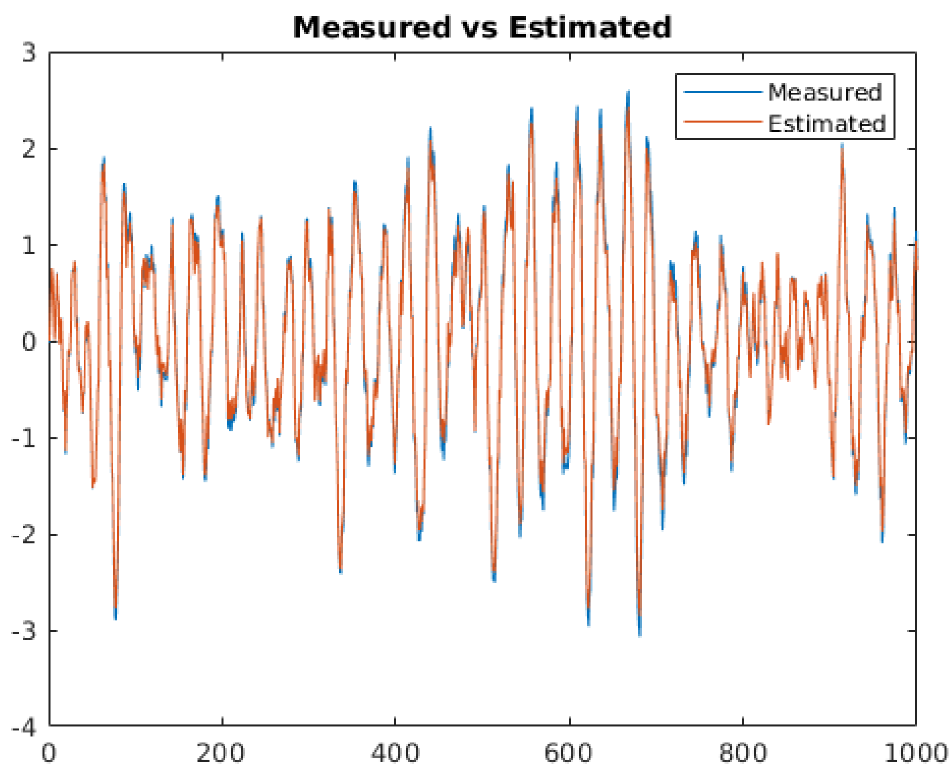
```
% Plot bode of ID and simulated TF
% Really good fit!
figure(4)
bode(true_tf);
hold on;
bode(est_tf);

% Plot Y and y on the same plot
% Really good fit, SNR 100:1
figure(5)
plot(1:1000, Y, 1:1000, est_y)
title("Measured vs Estimated")
legend("Measured", "Estimated")

% c. Repeat 10x with sigma = 0.01;
a_hist = [];
b_hist = [];
c_hist = [];
d_hist = [];

for i = 1:10
    u = randn(1000,1);
    y = dlsim(numd,dend,u);
    sigma = 0.01;
    Y = y + sigma * randn(1000,1);
    Y_ls = Y(3:end);
    H = [Y(2:end-1), Y(1:end-2), u(2:end-1), u(1:end-2)];
    est_coeff = H \ Y_ls;
    a_hist(i) = -est_coeff(1);
    b_hist(i) = -est_coeff(2);
    c_hist(i) = est_coeff(3);
    d_hist(i) = est_coeff(4);
end

% d. Repeat with sigma = 0.1, 1

sigmas = [0.01, 0.1, 1.0];
a_hist = zeros(10, length(sigmas));
b_hist = zeros(size(a_hist));
c_hist = zeros(size(a_hist));
d_hist = zeros(size(a_hist));

for i = 1:length(sigmas)
    sigma = sigmas(i);
    for j = 1:10
        u = randn(1000,1);
        y = dlsim(numd,dend,u);
        Y = y + sigma * randn(1000,1);

        Y_ls = Y(3:end);
        H = [Y(2:end-1), Y(1:end-2), u(2:end-1), u(1:end-2)];
        est_coeff = H \ Y_ls;

        a_hist(j, i) = -est_coeff(1);
```

```
        b_hist(j, i) = -est_coeff(2);
        c_hist(j, i) = est_coeff(3);
        d_hist(j, i) = est_coeff(4);
    end
    disp(['Stats for 10x, sigma = ', string(sigma)])
    disp(['a_mean: ', string(mean(a_hist(:,i)))]);
    disp(['b_mean: ', string(mean(b_hist(:,i)))]);
    disp(['c_mean: ', string(mean(c_hist(:,i)))]);
    disp(['d_mean: ', string(mean(d_hist(:,i)))]);

    disp(['a_std: ', string(std(a_hist(:,i)))]);
    disp(['b_std: ', string(std(b_hist(:,i)))]);
    disp(['c_std: ', string(std(c_hist(:,i)))]);
    disp(['d_std: ', string(std(d_hist(:,i)))]);

end

% Sigma = 0.01: Errors are pretty close to 0, stds are around 1/n *
% noise_sigma

% Sigma = 0.1: Errors are still pretty small, stds closer to
 noise_sigma

% Sigma = 1: Errors huge, std is basically equal to noise_sigma

% IMPLIES we are fitting to the noise instead of the model!
```

```
    "Stats for 10x, sigma = "    "0.01"

    "a_mean: "    "-1.8921"

    "b_mean: "    "0.94241"

    "c_mean: "    "0.24985"

    "d_mean: "    "-0.19785"

    "a_std: "    "0.0014031"

    "b_std: "    "0.0013335"

    "c_std: "    "0.00075019"

    "d_std: "    "0.00066744"

    "Stats for 10x, sigma = "    "0.1"

    "a_mean: "    "-1.406"

    "b_mean: "    "0.47608"

    "c_mean: "    "0.25001"

    "d_mean: "    "-0.076894"
```

```
"a_std: "    "0.038701"

"b_std: "    "0.037103"

"c_std: "    "0.0044208"

"d_std: "    "0.0098071"

"Stats for 10x, sigma = "    "1"

"a_mean: "    "-0.30899"

"b_mean: "    "-0.28661"

"c_mean: "    "0.23799"

"d_mean: "    "0.19936"

"a_std: "    "0.035601"

"b_std: "    "0.033429"

"c_std: "    "0.049734"

"d_std: "    "0.024945"
```


Bode Diagram
Magnitude (dB)
20
10
0
-10
-20
Phase (deg)
45
0
-45
-90
-135
-180
10^{-2}
10^{-1}
10^{0}
10^{1}
Frequency (rad/s)


Measured vs Estimated
Measured
Estimated
3
2
1
0
-1
-2
-3
-4
0
200
400
600
800
1000

Problem 5

```
%{
Justification of white noise for certain problems. Consider two
 problems:
(i) Simple first order low-pass filter with bandlimited white noise as
 the input:

y = G(s)*w, so that S_y(jw) = |G(jw)|^2 * S_w(jw), and the noise has
 the
PSD
S_1(w) = {A, |w| < w_c}, {0, |w| > w_c}
G(s) = 1 / (T_w*s + 1)

(ii) The same low pass system, but with pure white noise as the input.

S_1(w) = A \forall w
G(s) = 1 / (T_w*s + 1)

The first case seems quite plausible, but the second case has an input
 with infinite
variance and so is not physically realizable. However, the white noise
 assumption
simplifies the system analysis significantly, so it is important to
 see if the assumption
is justified. We test this with our two examples above:
a) Sketch the noise PSD and |G(jw)| for a reasonable value of T_w and
 w_c to
compare the two cases.
b) Determine the S_y(jw) for the two cases. Sketch these too.
c) Determine E[y^2] for the two cases.
d) Use these results to justify the following statement:
If the input spectrum is flat considerably beyond the system
 bandwidth,
there is little error introduced by assuming that the input spectrum
 is flat
out to infinity.
%}
```

Published with MATLAB® R2019b