Optimal - HW2

Walter Livingston

March 2024

Question I

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.

Part A

Find the math expression of the joint PDF function.

Solution

To begin solving for the mathematical expression for the joint PDF of a uniform circle with radius 2, the equation of a circle is needed.

$$(x - c_x)^2 + (y - c_y)^2 = r^2 \tag{1}$$

Given the above equation and the fact this circle is centered at the origin, the symbolic joint PDF is:

$$P_{x_1 x_2}(x_1, x_2) = \left\{ \begin{array}{ll} \frac{1}{\pi R^2}, & x_1^2 + x_2^2 \leq r^2 \\ 0, & x_1^2 + x_2^2 > r^2 \end{array} \right\} \tag{2}$$

Taking this symbolic equation and plugging in 2 for the radius, the joint pdf is:

$$P_{x_1 x_2}(x_1, x_2) = \left\{ \begin{array}{ll} \frac{1}{4\pi}, & x_1^2 + x_2^2 \leq 4 \\ 0, & x_1^2 + x_2^2 > 4 \end{array} \right\} \tag{3}$$

Part B

Find the conditional PDF $P_{x_2|x_1}(x_2|x_1 = 0.5)$?

Solution

Before performing the calculations to obtain the conditional PDF, the constraints on x_1 given x_2 need to be found and vice versa. These are as follows:

$$x_1 \leq \sqrt{4 - x_2^2} \tag{4}$$

$$x_2 \leq \sqrt{4 - x_1^2} \tag{5}$$

Given these, the bounds of x_2 given that $x_1 = 0.5$ can be found.

$$x_2 = \sqrt{4 - x_1^2}$$
$$x_2 = \sqrt{4 - 0.5^2}$$
$$x_2 = \sqrt{3.75} x_2 = \pm 1.9365$$

With these bounds calculated, the PDF of x_1 can be calculated.

$$P_{x_1}(x_1) = \int_{-\infty}^{\infty} P_{x_1 x_2}(x_1, x_2) dx_2$$
$$P_{x_1}(x_1) = \int_{-1.9365}^{1.9365} \frac{1}{4\pi}(x_1, x_2) dx_2$$
$$P_{x_1}(x_1) = \left.\frac{x_2}{4\pi}\right|_{x_2=1.9365} - \left.\frac{x_2}{4\pi}\right|_{x_2=-1.9365}$$
$$P_{x_1}(x_1) = 0.3082$$

Now with the PDF of x_1 solved for, the conditional PDF of $P_{x_2|x_1}(x_2|x_1 = 0.5)$ can be solved for.

$$P_{x_2|x_1}(x_2|x_1 = 0.5) = \frac{P_{x_1,x_2}(x_1, x_2)}{P_{x_1}(x_1 = 0.5)}$$
$$P_{x_2|x_1}(x_2|x_1 = 0.5) = \frac{\frac{1}{4\pi}}{0.3082}$$
$$P_{x_2|x_1}(x_2|x_1 = 0.5) = 0.2582$$

Part C

Are the two random variables uncorrelated?

Solution

To prove whether two random variables are uncorrelated, it needs to be proven that their covariance is 0 as shown below

$$E\left[(x_1 - \bar{x_1})(x_2 - \bar{x_2})^T\right] = 0 \tag{6}$$

First, the mean of the variables needs to be solved for.

$$E[x_1] = \int_{-\infty}^{\infty} x_1 P_{x_1}(x_1) dx_1$$
$$E[x_1] = \int_{-2}^{2} x_1 0.3082 dx_1$$
$$E[x_1] = \left.\frac{x_1^2}{4\pi}\right|_{x_1=2} - \left.\frac{x_1^2}{4\pi}\right|_{x_1=-2}$$
$$E[x_1] = 0$$

Given the symmetry of the joint PDF, it is clear that the solution above also applies to x_2. Now the covariances of x_1 and x_2 can be solved for.

$$E\left[(x_1)(x_2)^T\right] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x_1 x_2 P_{x_1,x_2}(x_1,x_2)dx_1dx_2$$

$$E\left[(x_1)(x_2)^T\right] = \int_{-2}^{2}\int_{-2}^{2} x_1 x_2 P_{x_1,x_2}(x_1,x_2)dx_1dx_2$$

$$E\left[(x_1)(x_2)^T\right] = \int_{-2}^{2}\int_{-2}^{2} x_1 x_2 \frac{1}{4\pi}dx_1dx_2$$

$$E\left[(x_1)(x_2)^T\right] = \left.\frac{x_1^2x_2^2}{4\pi}\right|_{x_1=2,x_2=2} - \left.\frac{x_1^2x_2^2}{4\pi}\right|_{x_1=-2,x_2=-2}$$

$$E\left[(x_1)(x_2)^T\right] = 0$$

Because the covariance of x_1 and x_2 is equal to 0, the two variables are uncorrelated.

Part D

Are the two random variables statistically independent?

Solution

To prove whether two random variables are independent, it needs to be proven that:

$$P_{x_1}(x_1)P_{x_2}(x_2) = P_{x_1,x_2}(x_1,x_2) \tag{7}$$

Plugging in the values calculated previously into the above equation yields:

$$(0.3082)(0.3082) \neq \frac{1}{4\pi}$$

Because these two are not equal, the two random variables are not independent.

Question II

The stationary process $x(t)$ has an autocorrelation function of the form:

$$R_x(\tau) = \sigma^2 e^{\beta|\tau|}$$

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.

Part A

What is the autocorrelation function for $y(t)$?

Solution

The autocorrelation function of $y(t)$ is calculated as follows:

$$
\begin{gathered}
R_y(\tau) = E\left[y(t)y(t+\tau)\right] = E\left[(ax(t)+b)(ax(t+\tau)+b)\right] \\
R_y(\tau) = E\left[a^2x(t)x(t+\tau) + abx(t) + abx(t+\tau) + b^2\right] \\
R_y(\tau) = a^2E\left[x(t)x(t+\tau)\right] + abE\left[x(t)\right] + abE\left[x(t+\tau)\right] + b^2 \\
R_y(\tau) = a^2R_x + ab\bar{x(t)} + ab\bar{x(t+\tau)} + b^2 \\
R_y(\tau) = a^2\sigma^2e^{\beta|\tau|} + 2ab\mu_x + b^2
\end{gathered}
$$

One note about this calculation is that because $x(t)$ is a stationary process, $E\left[x(t)\right] = E\left[x(t+\tau)\right] = \mu_x$.

Part B

What is the crosscorrelation function $R_{xy} = E\left[x(t)y(t+\tau)\right]$

Solution

The crosscorrelation function of $x(t)$ and $y(t)$ is calculated as follows:

$$
\begin{gathered}
R_{xy}(\tau) = E\left[x(t)y(t+\tau)\right] = E\left[x(t)ax(t+\tau) + bx(t)\right] \\
R_{xy}(\tau) = aR_x + b\mu_x \\
R_{xy}(\tau) = a\sigma^2e^{-\beta|\tau|} + b\mu_x
\end{gathered}
$$

Question III

Use least squares to identify a gyroscope scale factor (a) and bias (b). Simulate the gyroscope using:

$$
\begin{gathered}
g(k) = ar(k) + b + n(k) \\
n \sim N(0, \sigma = 0.3deg/s) \\
r(k) = 100sin(\omega t)
\end{gathered}
$$

Part A

Perform the least squares with 10 samples (make sure to pick ω so that you get one full cycle in 10 samples.)

Solution

To perform least squares, the above equations needs to be put into the form $y = Hx$. The measurement vector y is just $g(k)$. The state vector x is $\begin{bmatrix} a \\ b \end{bmatrix}$. Given these two vectors the H matrix becomes $\begin{bmatrix} r(k) & 1 \end{bmatrix}$. With the system in the correct form, Least Squares can now be performed. True values of $a = 1$ and $b = 0$ were chosen for this simulation. Also, a frequency of $0.6283 \frac{rad}{s}$ was chosen.

$$\hat{x} = (H^T H)^{-1} H^T y$$

With 10 samples, this yields $\hat{x} = \begin{bmatrix} 0.999 \\ -0.0121 \end{bmatrix}$.

Part 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.

Solution

A 1000 run Monte Carlo with 10 samples yields a numerical mean of $\bar{x} = \begin{bmatrix} 1 \\ -0.00465 \end{bmatrix}$ and a numerical standard deviation of $\sigma_x = \begin{bmatrix} 0.00128 \\ 0.0953 \end{bmatrix}$. The theoretical mean is just the true values of a and b being $\bar{x} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ which can be seen to match the numerical solution fairly closely. The theoretical standard deviation is calcualted as follows:

$$\sigma = \sqrt{\sigma_n^2 (H^T H)^{-1}} \tag{8}$$

where $sigma_n$ is the standard deviation on the noise. This yields an analytical standard deviation of $\sigma_x = \begin{bmatrix} 0.0013 \\ 0.0949 \end{bmatrix}$ which is very close to the numerical solution.

Part C

Repeat part (a) and (b) using 1000 samples. What does the theoretical and Monte Carlo standard deviation of the estimated errors approach?

Solution

A single Least Squares with 1000 samples yields $\hat{x} = \begin{bmatrix} 1 \\ 0.00103 \end{bmatrix}$. A 1000 run Monte Carlo with 1000 samples yields a numerical mean of $\bar{x} = \begin{bmatrix} 1 \\ 0.000186 \end{bmatrix}$ and a numerical standard deviation of $\sigma_x = \begin{bmatrix} 0.000139 \\ 0.00932 \end{bmatrix}$. The theoretical mean is still just the true values of a and b which do match the numerical solution very closely. The analytical standard deviation is calculated the same way as previously and yields $\sigma_x = \begin{bmatrix} 0.000134 \\ 0.0095 \end{bmatrix}$ which is again very close to the numerical. The standard deviations approach zeros given more and more samples.

Part 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.

Solution

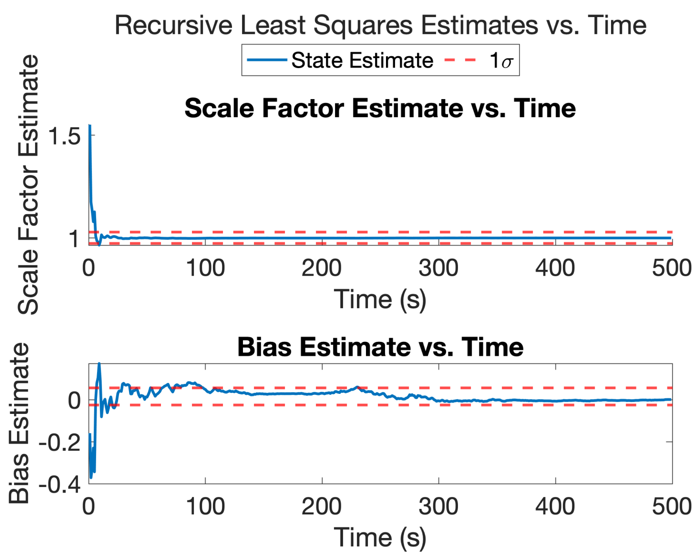

Figure 1: State Estimates from Recursive Least Squares

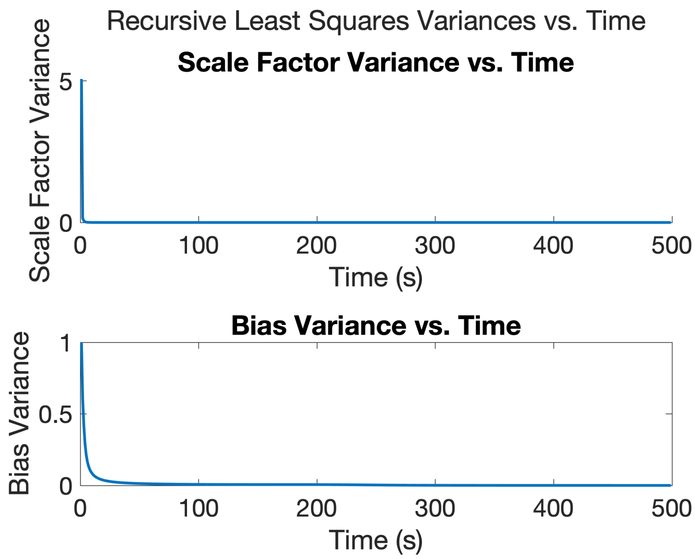

Figure 2: State Estimates from Recursive Least Squares

Question IV

Least Squares for System I.D. Simulate the following discrete system with a normal random input and output noise:

$$G(z) = \frac{0.25(z-0.8)}{z^2-1.90z+0.95}$$

Part A

Develop the H matrix for the least squares solution.

Solution

Before forming the H matrix and model of input to output given the coefficients of the transfer function is needed. This process is shown below:

$$G(z) = \frac{b_0 z - b_1}{z^2 - a_1 z + a_2}$$
$$y(k) + a_1 y(k-1) + a_2 y(k-2) = b_0 u(k-1) + b_1 u(k-2)$$
$$y(k) = -a_1 y(k-1) - a_2 y(k-2) + b_0 u(k-1) + b_1 u(k-2)$$
$$y(k+2) = -a_1 y(k+1) - a_2 y(k) + b_0 u(k+1) + b_1 u(k)$$
$$H = \begin{bmatrix} -y(k+1) & -y(k) & u(k+1) & u(k) \end{bmatrix}$$

Part 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.

Solution

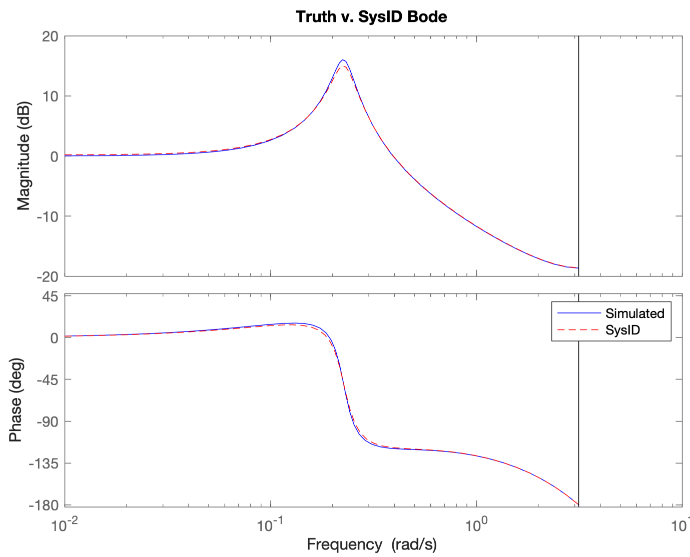

Figure 3: Bode Plot of True TF & SysID TF w/ $\sigma = 0.01$

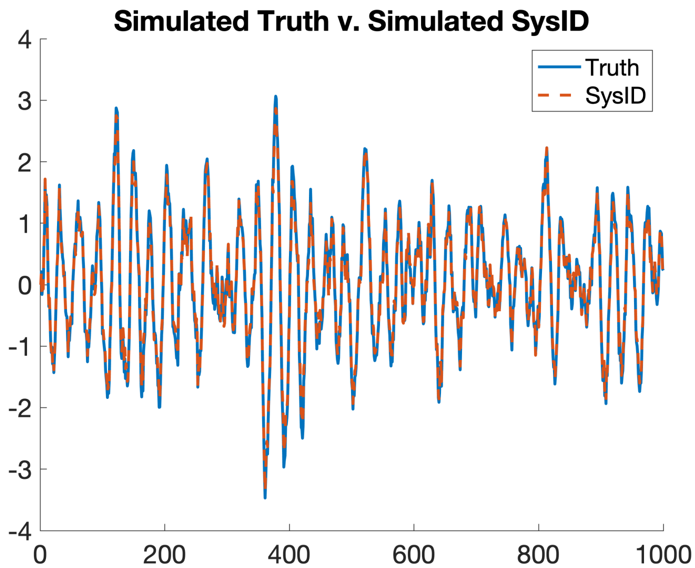

Figure 4: Plot of True TF & SysID TF w/ $\sigma = 0.01$

The fit of the data is fairly close. It can be seen that the system identification model is a good approximation of the real system. The SNR value for the system is 40.1726. This is good as the signal is much stronger than the noise of the system.

Part 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.

Solution

The 10-run Monte Carlo Simulation yields a mean $\bar{x} = \begin{bmatrix} -1.8922 \\ 0.9426 \\ 0.2497 \\ -0.1976 \end{bmatrix}$ and standard deviation $\sigma = \begin{bmatrix} 0.0011 \\ 0.0011 \\ \sim 0 \\ \sim 0 \end{bmatrix}$. The analytical equivalent mean is just the original parameters of transfer function. The analytical and numerical means appear to match fairly well, of course with a small amount of deviation. The analytical standard deviation is calculated as follows:

$$\sigma = \sigma_n (H^T H)^{-1} \tag{9}$$

This yields an analytical standard deviation of $\sigma = \begin{bmatrix} 0.0016 \\ 0.0016 \\ \sim 0 \\ \sim 0 \end{bmatrix}$. This is very nearly the same as the numerical solution.

Part D

Now use sigma between 0.1 and 1.0 and repeat parts b and c.

Solution

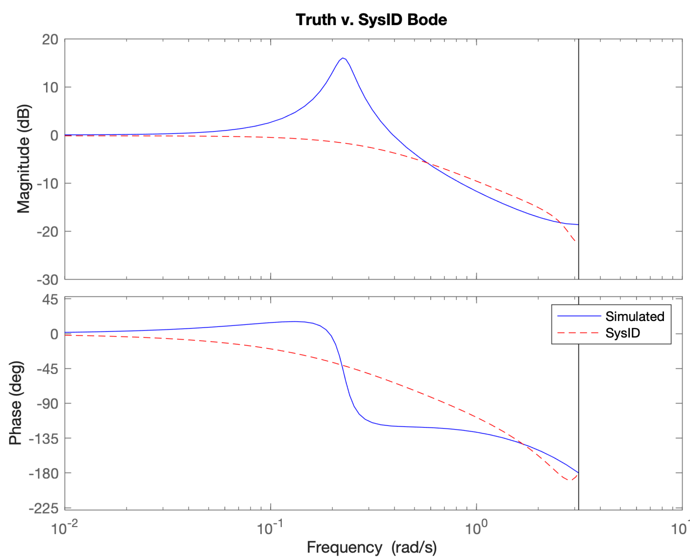

Figure 5: Bode Plot of True TF & SysID TF w/ $\sigma = 0.9$

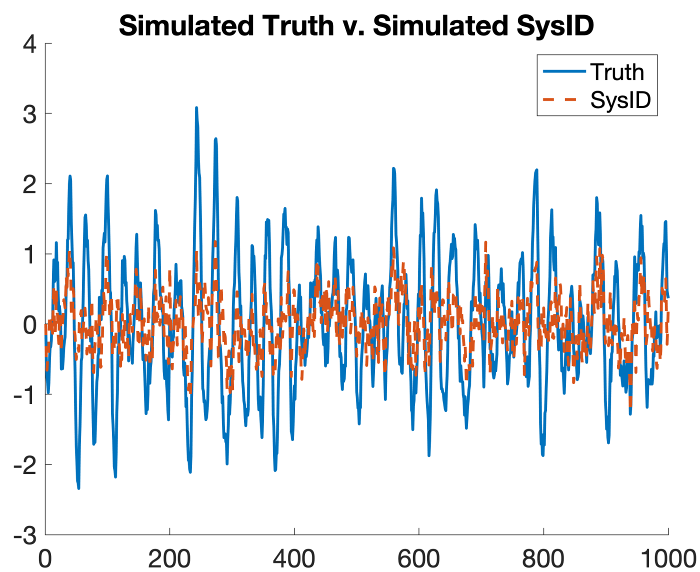

Figure 6: Plot of True TF & SysID TF w/ $\sigma = 0.9$

The fit is nowhere near as good as in Part B. Truly, it isn't even a valid representation of the system at this point. The SNR for this scenario is 1.3027 meaning the noise is a lot more prevalent in this case.
The Monte Carlo, utilizing the new σ value of 0.9, yields a numerical mean of $\bar{x} = \begin{bmatrix} -0.3208 \\ -0.2968 \\ 0.2659 \\ 0.1790 \end{bmatrix}$ which is not close to the original transfer function coefficients.

The numerical σ for this scenario is $\sigma = \begin{bmatrix} 0.0291 \\ 0.0209 \\ 0.0358 \\ 0.0285 \end{bmatrix}$. The analytical σ for the same scenario is $\sigma = \begin{bmatrix} 0.0260 \\ 0.0255 \\ 0.0287 \\ 0.0294 \end{bmatrix}$. While this does match the numerical solution's σ rather closely, the more important observation is that increasing the σ on the noise greatly increases the σ on the solution.

Part E

What can you conclude about using least squares for sys id with large amounts of noise?

Solution

Least Squares for system identification is good for low SNR systems, but it breaks down rather quickly with high SNR systems.

Question V

Justification of white noise for certain problems. Consider two problems: (i) Simple first order low-pass filter with bandlimited white noise as the input: $y = G(s)\omega$, so that $S_y(j\omega) = |G(j\omega)|^2 S_w(j\omega)$, and the noise has the PSD:

$$S_1(\omega) = \left\{ \begin{array}{ll} A, & |\omega| \leq \omega_c \\ 0, & |\omega| > \omega_c \end{array} \right\}$$
$$G(s) = \frac{1}{T_w s + 1}$$

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

$$S_2(\omega) = A, \forall \omega$$
$$G(s) = \frac{1}{T_w s + 1}$$

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.

Part A

Sketch the noise PSD and $|G(j\omega)|$ for a reasonable value of T_w and ω_c to compare the two cases.

Solution

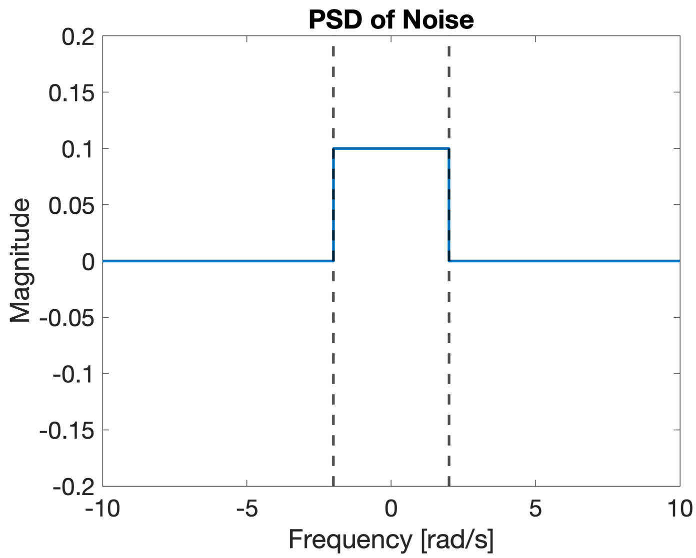

Figure 7: Noise PSD

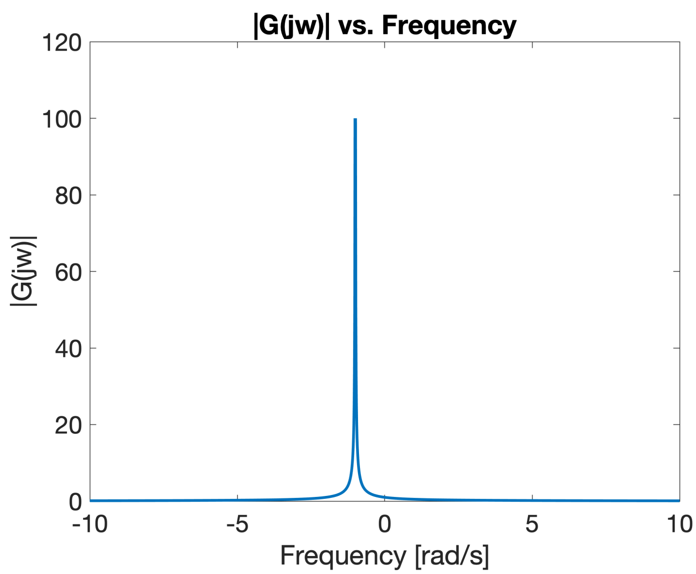

Figure 8: $|G(j\omega)|$

Part B

Determine the $S_y(j\omega)$ for the two cases. Sketch these too.

Solution

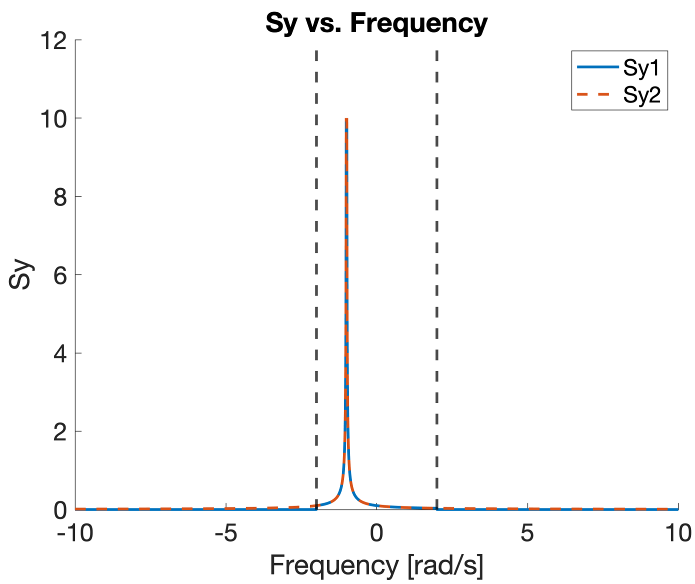

Figure 9: Sy

Part C

Determine $\left[y^2\right]$ for the two cases.

Solution

The equation for y is as follows:

$$y = G(s)w = \frac{w}{T_w s + 1} \tag{10}$$

where w is white noise. Given this, the expectation of y^2 is calculated as follows:

$$E\left[y^2\right] = E\left[\left(\frac{w}{T_w s+1}\right)\left(\frac{w}{T_w s+1}\right)\right]$$
$$E\left[y^2\right] = E\left[\frac{w^2}{T_w^2 s^2 + 2T_w s + 1}\right]$$
$$E\left[y^2\right] = \frac{1}{T_w^2 s^2 + 2T_w s + 1} E\left[w^2\right]$$
$$E\left[y^2\right] = \frac{\sigma_w^2}{T_w^2 s^2 + 2T_w s + 1} \sigma_y^2 = \frac{\sigma_w^2}{T_w^2 s^2 + 2T_w s + 1}$$

This is true due to the fact that w is zero-mean.

Part D

Use these results to justify the following statement: Îf 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.

Solution

Clearly from the above results the difference in bandlimited white noise, and pure, full-spectrum white noise is negligible. So the above assumption is valid.

Appendix A: MATLAB Code

```
%% PART 3
clear; close all; clc;

N1 = 10;                    % Number of Samples
Nmc = 1000;                 % Number of Monte Carlos
sigma_n = 0.3;              % Standard Deviation [deg/s]
t1 = 0:(N1-1);              % Time [s]
w1 = 2*pi/N1;               % Frequency [rad/s]
a = 1;                      % Gyroscope Bias Factor
b = 0;                      % Gyroscope Bias Factor

% Simulation
n1 = sigma_n*randn(N1,1);        % Noise
r1 = 100*sin(w1*t1)';            % Sine Wave
g1 = a*r1 + b*ones(N1,1) + n1;   % Gyroscope Measurements

```

```
% (A)
% Least Squares
Ha = [r1 ones(N1,1)];          % Geometry Matrix
xa = pinv(Ha)*g1;              % State Estimate
fprintf('a) 10 Sample; Individual Least Squares Estimate: [%0.3g %0.3g]\n', xa);

% (B)
xb = zeros(2,Nmc);
for i = 1:Nmc
    n1 = sigma_n*randn(N1,1);         % Noise
    g1 = a*r1 + b*ones(N1,1) + n1;  % Gyroscope Measurements
    Hb = [r1 ones(N1,1)];             % Geometry Matrix
    xb(:,i) = pinv(Hb)*g1;            % State Estimate
end
sigmab = std(xb,0,2);         % Sample Sigma
meanb = mean(xb,2);           % Sample Mean
sigmab_an = sqrt(sigma_n^2*inv(Hb'*Hb));
fprintf('b) 10 Sample; Monte Carlo Least Squares Mean: [%0.3g %0.3g]\n', meanb);
fprintf('b) 10 Sample; Monte Carlo Least Squares Sigma: [%0.3g %0.3g]\n', sigmab);

N2 = 1000;
t2 = 0:(N2-1);                % Time [s]
w2 = 2*pi/N2;                 % Frequency [Hz]

% Simulation
n2 = sigma_n*randn(N2,1);         % Noise
r2 = 100*sin(w2*t2)';             % Sine Wave
g2 = a*r2 + b*ones(N2,1) + n2;  % Gyroscope Measurements

% (C)
% Least Squares
Hc = [r2 ones(N2,1)];          % Geometry Matrix
xc = pinv(Hc)*g2;              % State Estimate
fprintf('c) 1000 Sample; Individual Least Squares Estimate: [%0.3g %0.3g]\n', xc);

xc = zeros(2,Nmc);
for i = 1:Nmc
    n2 = sigma_n*randn(N2,1);         % Noise
    g2 = a*r2 + b*ones(N2,1) + n2;  % Gyroscope Measurements
```

```
    Hc = [r2 ones(N2,1)];           % Geometry Matrix
    xc(:,i) = pinv(Hc)*g2;          % State Estimate
end
sigmac = std(xc,0,2);        % Sample Sigma
meanc = mean(xc,2);          % Sample Mean

sigmac_an = sqrt(sigma_n^2*inv(Hc'*Hc));
fprintf('c) 1000 Sample; Monte Carlo Least Squares Mean: [%0.3g %0.3g]\n', meanc);
fprintf('c) 1000 Sample; Monte Carlo Least Squares Sigma: [%0.3g %0.3g]\n', sigmac);

% (D)
N3 = 1000;
batch = 2;
t3 = 0:(N3-1);               % Time [s]
w3 = 2*pi/N3;                % Frequency [Hz]
Rk = diag([sigma_n, sigma_n]);

% Simulation
n3 = sigma_n*randn(N3,1);        % Noise
r3 = 100*sin(w3*t3)';            % Sine Wave
g3 = a*r3 + b*ones(N3,1) + n3;  % Gyroscope Measurements

x3 = zeros(2,(N3/batch)-1);
P3 = zeros(2,2,(N3/batch)-1);

H3 = [r3 ones(N3,1)];

x3(:,1) = pinv(H3(1:batch,:))*g3(1:batch);
P3(:,:,1) = inv(H3(1:batch,:)'*H3(1:batch,:));

for i = 2:(N3/batch)-1
    start = i*batch; stop = start + batch - 1;
    Hk = H3(start:stop, :);
    K = P3(:,:,i-1)*Hk'*inv(Hk*P3(:,:,i-1)*Hk' + Rk);
    P3(:,:,i) = (eye(2) - K*Hk)*P3(:,:,i-1);
    x3(:,i) = x3(:,i-1) + K*(g3(start:stop) - Hk*x3(:,i-1));
end

sigma3 = std(x3,0,2);
mean3 = mean(x3,2);

figure();
```

```
t = tiledlayout(2,1);
title(t, 'Recursive Least Squares Estimates vs. Time', 'FontSize', 20);
nexttile(t);
hold('on');
plot(x3(1,:), 'LineWidth', 2);
yline(sigma3(1) + mean3(1), '--r', 'LineWidth', 2);
yline(-sigma3(1) + mean3(1), '--r', 'LineWidth', 2);
xlabel('Time (s)');
ylabel('Scale Factor Estimate');
title('Scale Factor Estimate vs. Time');
ax = gca;
ax.FontSize = 18;

nexttile(t);
plot(x3(2,:), 'LineWidth', 2);
yline(sigma3(2) + mean3(2), '--r', 'LineWidth', 2);
yline(-sigma3(2) + mean3(2), '--r', 'LineWidth', 2);
xlabel('Time (s)');
ylabel('Bias Estimate');
title('Bias Estimate vs. Time');
ax = gca;
ax.FontSize = 18;

leg = legend('State Estimate', '1\sigma', 'Orientation', 'horizontal');
leg.Layout.Tile = 'north';
ax = gca;
ax.FontSize = 18;

exportgraphics(gcf, 'figures/p3d_state.png', 'Resolution', 300);

figure();
t = tiledlayout(2,1);
title(t, 'Recursive Least Squares Variances vs. Time', 'FontSize', 20);
nexttile(t);
hold('on');
plot(squeeze(P3(1,1,:)), 'LineWidth', 2);
xlabel('Time (s)');
ylabel('Scale Factor Variance');
title('Scale Factor Variance vs. Time');
ax = gca;
ax.FontSize = 18;

```

```
nexttile(t);
plot(squeeze(P3(2,2,:)), 'LineWidth', 2);
xlabel('Time (s)');
ylabel('Bias Variance');
title('Bias Variance vs. Time');
ax = gca;
ax.FontSize = 18;

exportgraphics(gcf, 'figures/p3d_variance.png', 'Resolution', 300);

%% PART 4
clear; close all; clc;

N = 1000;                          % Number of Samples

numd = 0.25*[1 -0.8];              % Discrete TF Numerator
dend = [1 -1.9 0.95];              % Discrete TF Denominator
tfd = tf(numd, dend, -1);              % Discrete TF
u = randn(N,1);                    % Input Noise
y = dlsim(numd,dend,u);            % Output Simulation
sigma1 = 0.01;                     % Output Noise Standard Deviation
Y = y + sigma1*randn(N,1);         % Output w/ Noise

Ha = [-Y(2:end-1) -Y(1:end-2) u(2:end-1) u(1:end-2)];
xa = pinv(Ha)*Y(3:end);

numID = xa(3:4)';
denID = [1 xa(1:2)'];
tfID = tf(numID, denID, -1);
yID = dlsim(numID,denID,u);
snr1 = snr(y, Y-y);

figure();
bode(tfd, '-b');
hold('on');
bode(tfID, '--r')
title('Truth v. SysID Bode');
legend('Simulated', 'SysID');

exportgraphics(gcf, 'figures/p4b_bode.png', 'Resolution', 300);

```

```
figure();
hold('on');
plot(y, 'LineWidth', 2);
plot(yID, '--', 'LineWidth', 2);
title('Simulated Truth v. Simulated SysID');
legend('Truth', 'SysID');
ax = gca;
ax.FontSize = 18;

exportgraphics(gcf, 'figures/p4b_tf.png', 'Resolution'
    , 300);

% snrVal = snr(y, Y - y)

N_mc = 10;
x_mc1 = zeros(4,N_mc);
for i = 1:N_mc
    u = randn(N,1);
    y = dlsim(numd,dend,u);
    Y = y + sigma1*randn(N,1);
    Ha = [-Y(2:end-1) -Y(1:end-2) u(2:end-1) u(1:end
        -2)];
    x_mc1(:,i) = pinv(Ha)*Y(3:end);
end
sigma_x_an1 = sqrt(diag(sigma1.^2*inv(Ha'*Ha)));

sigma_x_mc1 = std(x_mc1,1,2);
mean_x_mc1 = mean(x_mc1,2);

sigma2 = 0.9;
Y = y + sigma2*randn(N,1);       % Output w/ Noise

Hd = [-Y(2:end-1) -Y(1:end-2) u(2:end-1) u(1:end-2)];
xd = pinv(Hd)*Y(3:end);

numID = xd(3:4)';
denID = [1 xd(1:2)'];
tfID = tf(numID, denID, -1);
yID = dlsim(numID,denID,u);
snr2 = snr(y, Y-y);

figure();
bode(tfd, '-b');
hold('on');
bode(tfID, '--r')
title('Truth v. SysID Bode');
```

```
legend('Simulated', 'SysID');

exportgraphics(gcf, 'figures/p4d_bode.png', 'Resolution', 300);

figure();
hold('on');
plot(y, 'LineWidth', 2);
plot(yID, '--', 'LineWidth', 2);
title('Simulated Truth v. Simulated SysID');
legend('Truth', 'SysID');
ax = gca;
ax.FontSize = 18;

exportgraphics(gcf, 'figures/p4d_tf.png', 'Resolution', 300);

x_mc2 = zeros(4,N_mc);
for i = 1:N_mc
    u = randn(N,1);
    y = dlsim(numd,dend,u);
    Y = y + sigma2*randn(N,1);
    Hd = [-Y(2:end-1) -Y(1:end-2) u(2:end-1) u(1:end-2)];
    x_mc2(:,i) = pinv(Hd)*Y(3:end);
end
sigma_x_an2 = sqrt(diag(sigma2^2*inv(Hd'*Hd)));

sigma_x_mc2 = std(x_mc2,1,2);
mean_x_mc2 = mean(x_mc2,2);

%% PART V
clear;
Tw = 1;
A = 0.1;
wc = 2;      % [rad/s]
w1 = -10:0.01:10; % [rad/s]

syms w
Sw = piecewise(abs(w) <= wc, A, abs(w) > wc, 0);
G = abs(1./(Tw*w1 + 1));

for i = 1:length(w1)
    if abs(w1(i)) < wc
        Sy1(i) = (A)/(Tw*w1(i) + 1);
    else
```

```
        Sy1(i) = 0;
    end
end

Sy2 = A./(Tw.*w1 + 1);

figure();
fplot(Sw, [-10 10], 'LineWidth', 2);
xlabel('Frequency [rad/s]');
ylabel('Magnitude');
title('PSD of Noise');
xline(-wc, '--k','LineWidth', 2);
xline(wc, '--k','LineWidth', 2);
ylim([-A*2 A*2])
ax = gca;
ax.FontSize = 18;

exportgraphics(gcf, 'figures/p5_noise_psdl.png', '
   Resolution', 300);

figure();
plot(w1, G, 'LineWidth', 2);
xlabel('Frequency [rad/s]');
ylabel('|G(jw)|');
title('|G(jw)| vs. Frequency');
ax = gca;
ax.FontSize = 18;

exportgraphics(gcf, 'figures/p5_gjw.png', 'Resolution'
   , 300);

figure();
hold('on');
plot(w1,abs(Sy1), 'LineWidth', 2);
plot(w1,abs(Sy2), '--', 'LineWidth', 2);
xline(-wc, '--k','LineWidth', 2);
xline(wc, '--k','LineWidth', 2);
xlabel('Frequency [rad/s]');
ylabel('Sy');
title('Sy vs. Frequency');
legend('Sy1', 'Sy2');
ax = gca;
ax.FontSize = 18;

exportgraphics(gcf, 'figures/p5_sy.png', 'Resolution',
    300);
```