Optimal HW3

Walter Livingston

April 2024

Note: All problems are to be sampled at 10 Hz (Ts=0.1) except for problem #3.

Question I

Kalman Filter at its best – simulation (actually the Kalman filter is also quite reliable when we have an excellent model and low noise sensors). Suppose we have a 2 nd order system that we are regulating about zero (position and velocity) by wrapping an "optimal" control loop around the system. The new dynamics of the continuous time system are given by the closed-loop A matrix:

$$A_{CL} = \begin{bmatrix} 0 & 1 \\ -1 & -1.4 \end{bmatrix} \tag{1}$$

Suppose our measurement is simply position ($C = \begin{bmatrix} 1 & 0 \end{bmatrix}$). There is a white noise process disturbance (force, $B_w = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$) acting on the controlled system.

Part A

Simulate the controlled system with the disturbance force ($1\sigma = 1$) for 100 seconds at 10Hz sample rate.

Solution

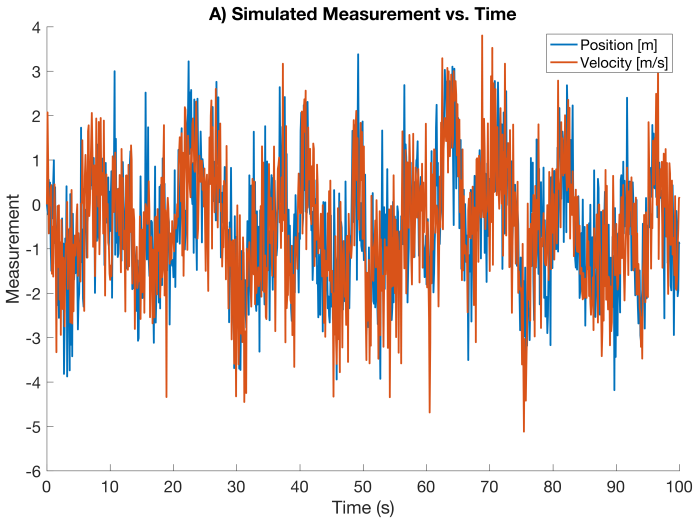

Figure 1: Simulated Measurements vs. Time

Part B

What is Q, Q_d, R_d?

Solution

$$Q_c = 4$$

$$Q_d = \int B_w Q_c {B_w}^T d\tau = 0.4$$

$$R_d = 1$$

Part C

Calculate the steady state Kalman gain for the system. This can be done in one of many ways: iterate the kalman filter until it converges, dlqe.m, dare.m, kalman.m, dlqr.m (+ predictor to current estimator trick), etc. What is the steady state covariance of the estimates after the time update, $P^{(-)}$, as well as after the measurement update, $P^{(+)}$. Where are the poles of the estimator?

Solution

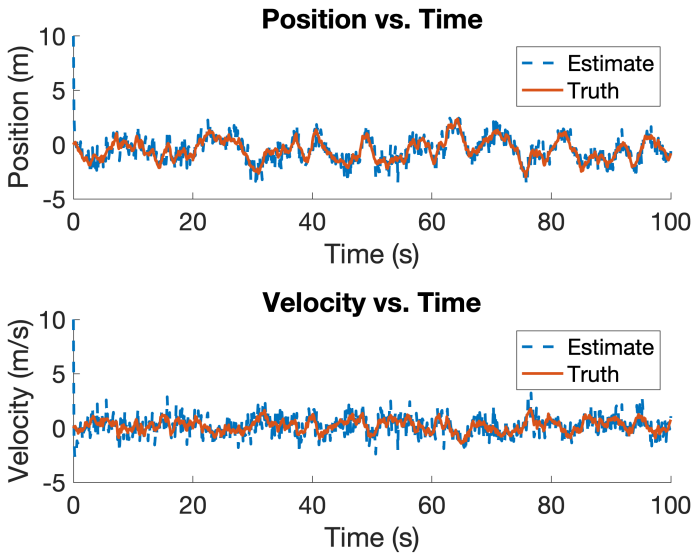

Figure 2: State Estimate vs. Time

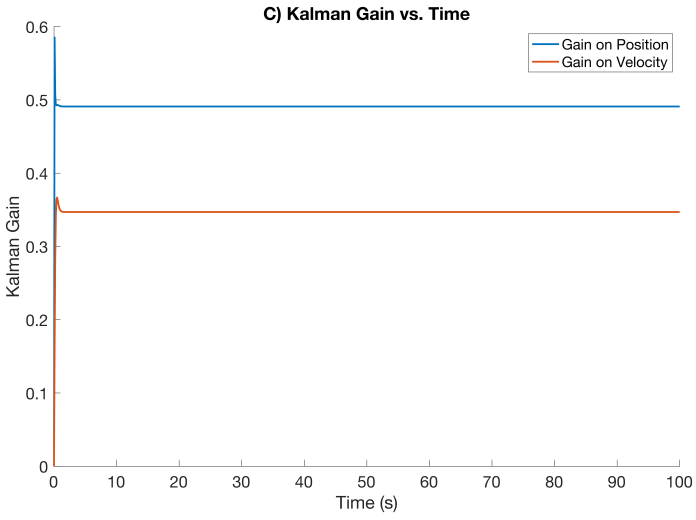

Figure 3: Kalman Gain vs. Time

$$L_{ss} = \begin{bmatrix} 0.4910 \\ 0.3479 \end{bmatrix}$$

$${P_{ss}}^{-} = \begin{bmatrix} 0.9646 & 0.6818 \\ 0.6818 & 0.6538 \end{bmatrix}$$

$${P_{ss}}^{+} = \begin{bmatrix} 0.4910 & 0.3470 \\ 0.3470 & 0.4172 \end{bmatrix}$$

$$s = 0.6845 \pm j0.1179$$

Part D

Now use the steady state kalman filter to generate an estimate ($\hat{x}$ and $\hat{\dot{x}}$) of the 2 states over time. Calculate the norm of the standard deviation of the errors for each state.

$$N = \sqrt{(std(\dot{x} - \hat{\dot{x}}))^2 + std(x - \hat{x})^2} \tag{2}$$

Solution

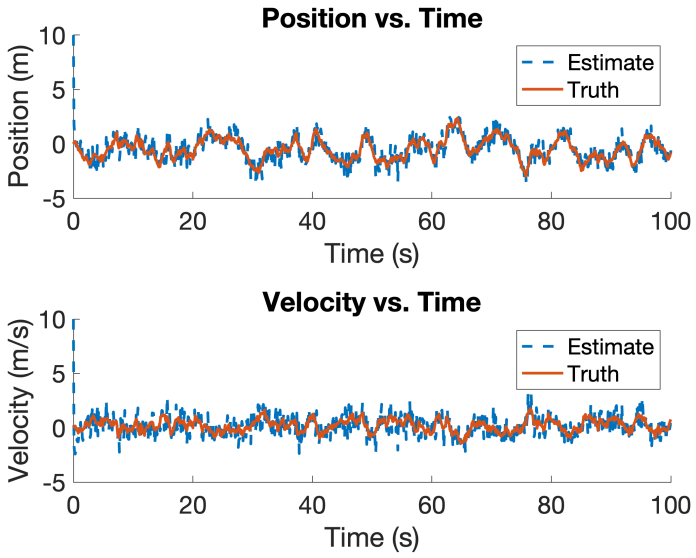

Figure 4: SS Kalman Filter State Estimate vs. Time

$$N = 1.0816$$

$$N_{ss} = 1.0788$$

Part E

Change the ratio of the Q and R_d weights in the Kalman filter design (and repeat part d with the new Kalman gain but DO NOT regenerate a new x and $\dot{x}$) and determine the effect on the estimation errors. For what ratio of Q and R_d are the errors minimized? Note: Often in practice we do not know the actual Q and R_d so these tend to be "tuning" parameters we use to tune our filter. However, according to Kalman, the estimation errors are only minimized if we use the Q and R_d of the physical system.

Solution

It appears the norm of the estimate errors are minimized when $Q_c = 4$ and $R_d = 1$ which makes sense as that is the systems true noise values and Kalman stated it was at the system true values that the filter was optimal.

Question II

Download the data hw3_2 from the website. The data is in the form $\begin{bmatrix} t & y \end{bmatrix}$. Suppose we want to design an estimator to estimate the bias in the measurement y. We believe the bias (x) is constant so we use the model given by:

$$\dot{x} = 0 \tag{3}$$

$$y_k = x_k + \nu_k \tag{4}$$

$$\nu_k \sim N(0, 1) \tag{5}$$

Part A

Run the Kalman filter estimator with $Q_d = 0$. What happnes at $t > 100$ seconds? Why? Calculate the steady state Kalman gain L_{ss}. Plot $L(k)$. This is known as the filter "going to sleep" (becomes a least squares estimator).

Solution

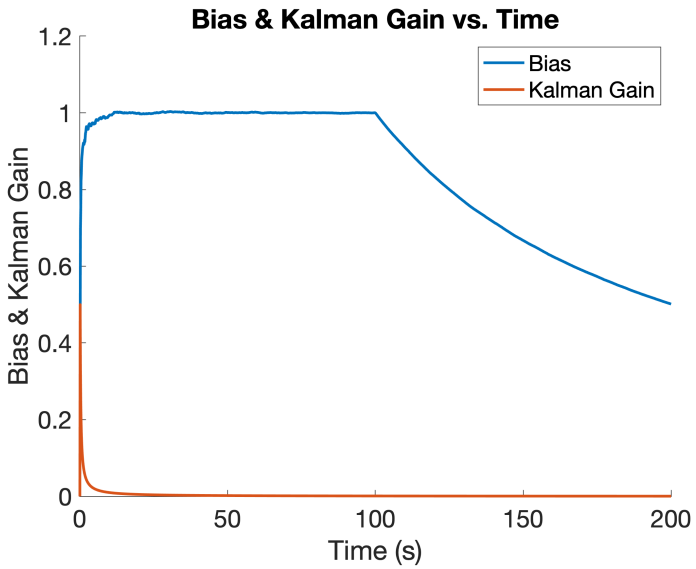

Figure 5: Bias & Kalman Gain vs. Time

At $t > 100$, the estimate of the bias slowly starts to decrease. The actual bias has a step change in it's value and the Kalman Filter is trying to estimate that. The Kalman Filter has "gone to sleep" though, and is no longer accurately estimating the state. This can be inferred from the steady-state value of the Kalman gain:

$$L_{ss} = 0$$

Part B

To offset this problem we will "tune" Q_d to track the bias. What is the effect of changing Q_d on the ability to track the step change in the bias? (try values of Q_d from 0.0001 to 0.01 and plot $L(k)$ as well as the estimate of the bias ($\hat{x}$)). What is the tradeoff?

Solution

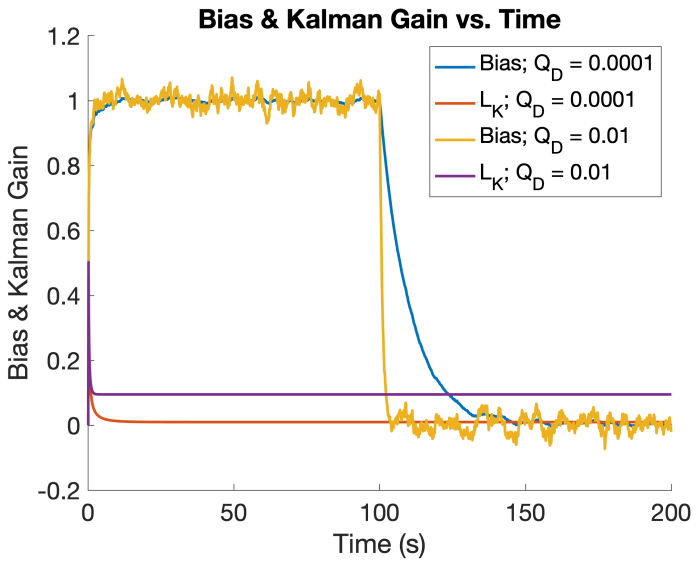

Figure 6: Bias & Kalman Gain vs. Time

The tradeoff with higher Q_d values is better/"faster" tracking of the estimate, though by gaining more noise on the estimate.

Part C

Now filter the measurement using the first order low-pass filter:

$$H(z) = \frac{sqrtQ_d}{z - (1 - sqrtQ_d)} \tag{6}$$

us ethe command: yf = filter(numd, dend, y, y0)

Solution

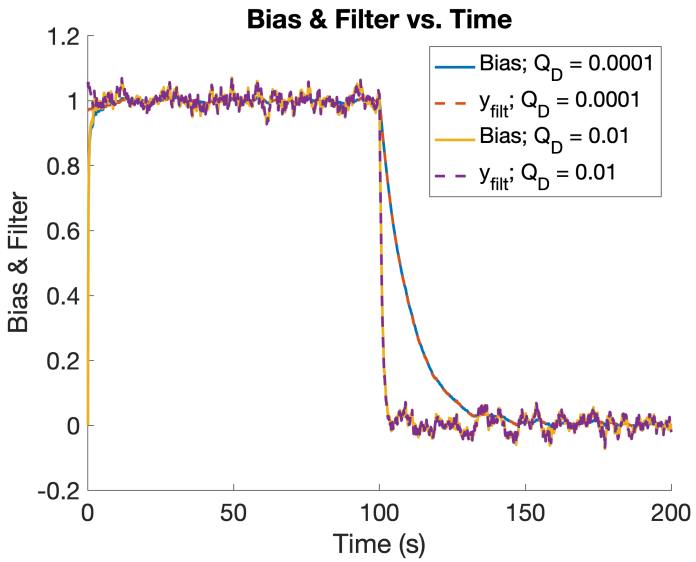

Figure 7: Bias & Filter vs. Time

Part D

How does this compare to the Kalman filter solution. Why are these two filters the same for this problem?

Solution

The filtered signal and the Kalman Filter estimate match exactly. The Kalman Filter for this setup has no time update. It is just using measurements (y) to filter the process noise Q_d which leaves the bias. This is exactly what the provided filter is doing as well.

Question III

Design a "Navigation" type Kalman filter to estimate the states [East, North, Radar_Bias, Psi, Gyro_Bias]. (Note: this is a non-linear problem that requires an Extended Kalman Filter (EKF) to do correctly). However we can solve the problem in one of two ways: (i) linearize the equations about the nominal operating point and produce a constant A matrix for that operating point, or (ii) simply update the A matrix at every time step with our measurements or estimates. Download the data hw3_3 from the website and run the filter sampled at 5 Hz

Part A

How did you choose the covariance values for Q_d (especially for the radar and gyro biases).

Solution

Before beginning to form the state-space model of the filter the states being estimated needed to be identified.

$$x = \begin{bmatrix} E \\ N \\ \psi \\ b_r \\ b_g \end{bmatrix}$$

With the states identified, equations relating the measurements to those states need to be identified.

$$y = \begin{bmatrix} \tilde{E} \\ \tilde{N} \\ \tilde{\psi} \\ \dot{\tilde{\psi}} \\ \tilde{V} \end{bmatrix}$$

$$\tilde{E} = E + \nu_E$$
$$\tilde{N} = N + \nu_N$$
$$\tilde{\psi} = \psi + \nu_\psi$$
$$\dot{\tilde{\psi}} = \dot{\psi} + b_g + \omega_g$$
$$\tilde{V} = V + b_r + \omega_r$$

With the states and measurement equations identified, the last step before forming the state-space representation is to identify equations for the state derivatives.

$$\dot{E} = V\sin(\psi) = (\tilde{V} - b_r - \omega_r)\sin(\psi)$$
$$\dot{N} = V\cos(\psi) = (\tilde{V} - b_r - \omega_r)\cos(\psi)$$
$$\dot{\psi} = \dot{\psi} = \dot{\tilde{\psi}} - b_g - \omega_g$$
$$\dot{b_r} = 0$$
$$\dot{b_g} = 0$$

With all the equations identified, the state space representation can start to take form. This begins with identifying $f(x)$ and $h(x)$ and taking their Jacobians.

$$\dot{x} = f(x) = \begin{bmatrix} (\tilde{V} - b_r)\sin(\psi) \\ (\tilde{V} - b_r)\cos(\psi) \\ \dot{\tilde{\psi}} - b_g \\ 0 \\ 0 \end{bmatrix} \quad y = h(x) = \begin{bmatrix} E \\ N \\ \psi \\ \dot{\psi} + b_g \\ V + b_r \end{bmatrix}$$

$$A = \frac{\partial f}{\partial x} = \begin{bmatrix} 0 & 0 & (\tilde{V} - b_r)\cos(\psi) & -\sin(\psi) & 0 \\ 0 & 0 & (\tilde{V} - b_r)\cos(\psi) & -\cos(\psi) & 0 \\ 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$H = \frac{\partial h}{\partial x} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Now with all equations and Jacobians identified and solved, the normal Kalman Filter techniques can be applied. The results are presented below.

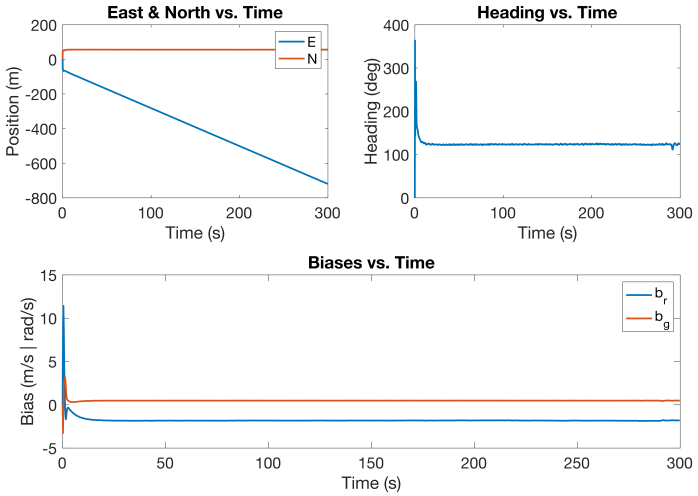

Figure 8: Nav KF States vs. Time

Q_d and R_d were both tuned (starting at values of 1) until the estimate was consistent and noise was minimized on the estimates. The matrices are as

follows:

$$Q_d = \begin{bmatrix} 0.01 & 0 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 & 0 \\ 0 & 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0 & 0.001 & 0 \\ 0 & 0 & 0 & 0 & 0.001 \end{bmatrix}$$

$$R_d = \begin{bmatrix} 0.1 & 0 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 & 0 \\ 0 & 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0 & 0.1 \end{bmatrix}$$

Part B

How does the bias estimation compare to the Least Squares Solution. How does the bias estimate compare to the Recursive Least Squares solution if you make the covariance (Q_d) of the bias estimates equal to zero.

Solution

The Recursive Least Squares solution used the same observation matrix presented in Part A.

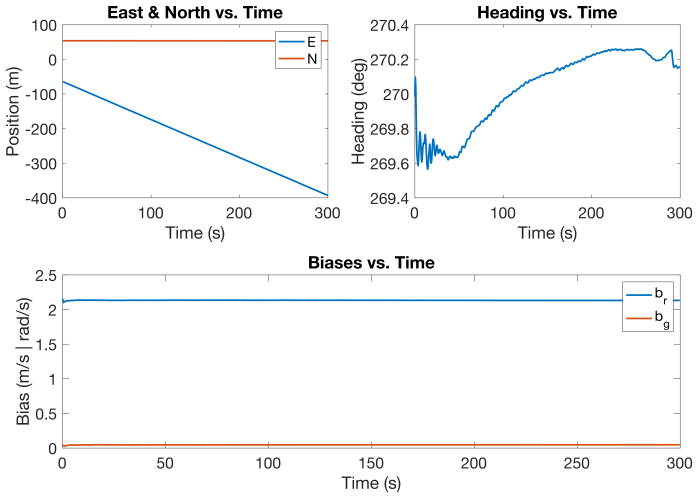

Figure 9: Nav RLS States vs. Time

The bias estimates from the Recursive Least Squares (RLS) estimator match very closely with the Kalman Filter (KF) bias estimates. It should be noted that the heading estimate is of a small scale, and the initial state "guess" isn't

zero making it look noisier. Looking at the plot scale shows this is a small magnitude of variation.

Part C

Integrate the last 40 seconds of data to see how well you have estimated the biases. This can simply be done by "turning off" the measurements in the observation matrix! Why do the bias estimates remain constant during the 40 second "outage?"

Solution

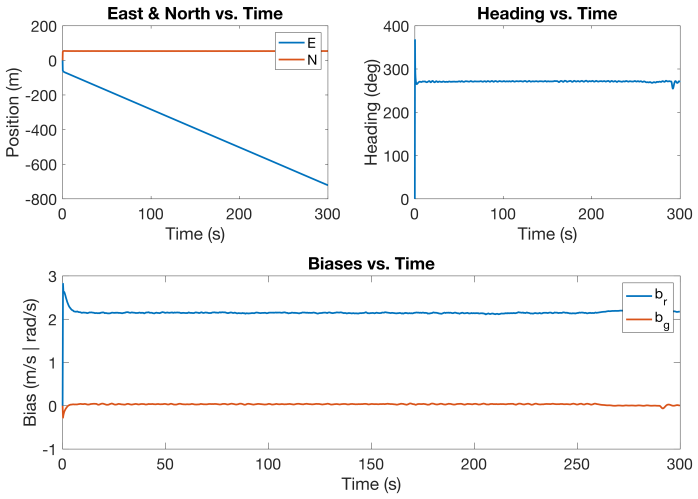

Figure 10: Nav KF States w/ 40s "Outage" vs. Time

The biases remain fairly constant during the "outage" as they have no time update, and are now not being updated by the measurements either.

Question IV

Estimate for Vehicle Dynamics. The yaw dynamics of a car (for a stability control system) can be described by the following model (at 25 m/s):

$$\dot{X} = \begin{bmatrix} -2.62 & 12 \\ -0.96 & -2 \end{bmatrix} \begin{bmatrix} \dot{\Psi} \\ \beta \end{bmatrix} + \begin{bmatrix} 14 \\ 1 \end{bmatrix} \delta \tag{7}$$

$$y_k = \begin{bmatrix} 1 & 0 \end{bmatrix} X_k + \nu_k \tag{8}$$

where:

$$\dot{\Psi} - VehicleYawRate \tag{9}$$
$$\beta - VehicleSideSlipAngle \tag{10}$$
$$\delta - SteerAngle \tag{11}$$
$$\nu_k - SampleSensorNoise \tag{12}$$

Part A

Assuming we can only measure the yaw rate ($\nu_k \sim N(0, 0.1^2)$), design a Kalman filter to do full state estimation (select a reasonable Q_d). Provide a unit step stee input and estimate both states. On one page plot the actual states and estimated states (use subplot(2,2,n) for each of the two states). Where are the steady state poles of the estimator?

Solution

Below is a plot of the simulated system yaw-rate vs. the simulated measurement thereof.

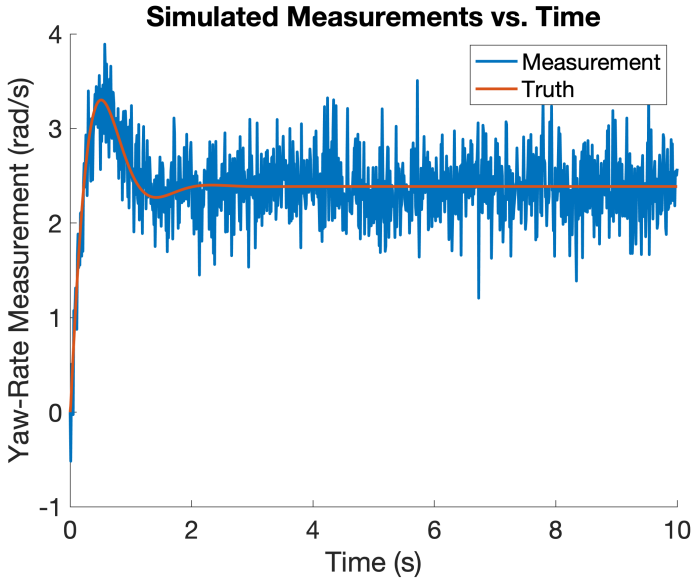

Figure 11: Simulated Measurement vs. Time

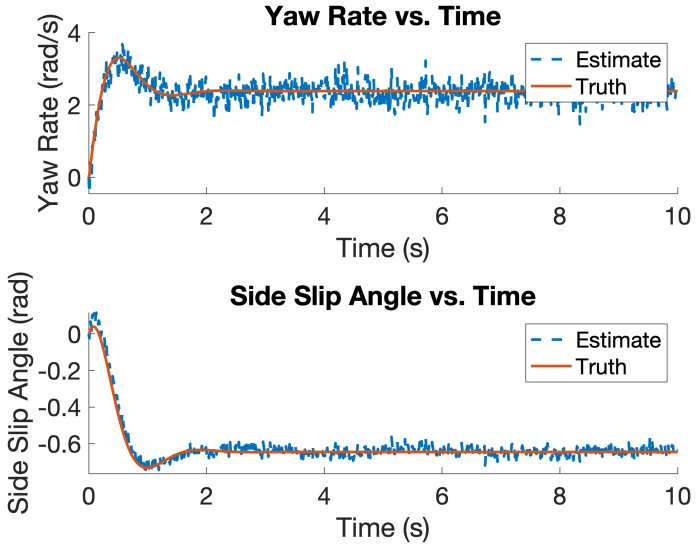

Figure 12: State Estimates & Truth vs. Time

The yaw-rate is estimated fairly well, which is to be expected as we have a direct measurement of the state as well as a model for it. The side-slip angle estimates well after a significant amount of tuning. The process noise on that term has to be turned down significantly. The resulting Q_d is as follows:

$$Q_d = \begin{bmatrix} 1 & 0 \\ 0 & 0.03 \end{bmatrix}$$

Part B

Now somebody has loaded the trunk of the vehicle with bricks, changing the CG of the vehicle so now the actual model (at 25 m/s) is:

$$\dot{X} = \begin{bmatrix} -2.42 & 4 \\ -0.99 & -2 \end{bmatrix} \begin{bmatrix} \dot{\Psi} \\ \beta \end{bmatrix} + \begin{bmatrix} 18 \\ 1 \end{bmatrix} \delta \tag{13}$$

$$y_k = \begin{bmatrix} 1 & 0 \end{bmatrix} X_k + \nu_k \tag{14}$$

NOTE: We do not know that somebody has loaded the trunk and that the C.G. has changed, therefore we must use the model for part a in our Kalman Filter. Redo part a. Can you estimate the slip angle correctly? Try various Q_d.

Solution

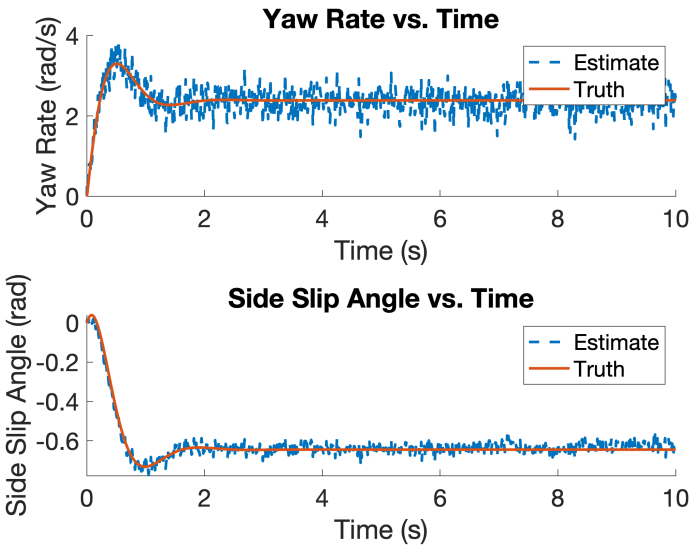

Figure 13: State Estimates & Truth vs. Time

The slip angle can be estimated correctly given that the process noise on the slip angle is double from Part A. The same was done to the process noise on the yaw-rate. The resulting Q_d is as follows:

$$Q_d = \begin{bmatrix} 2 & 0 \\ 0 & 0.06 \end{bmatrix}$$

Part C

Now lets say we have a noisy measurement of the slip angle ($\eta_k \sim N(0, 0.5^2)$):

$$y_k = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} X_k + \begin{bmatrix} \nu_k \\ \eta_k \end{bmatrix} \tag{15}$$

Assuming the sensor noises are uncorrelated, what is R?

Solution

Part D

Redo part a. What is the effect of changing the element Q_d associated with the slip angle estimate. What must Q_d equal to ensure an unbiased estimate of the states (How much filtering does that provide)?

Solution

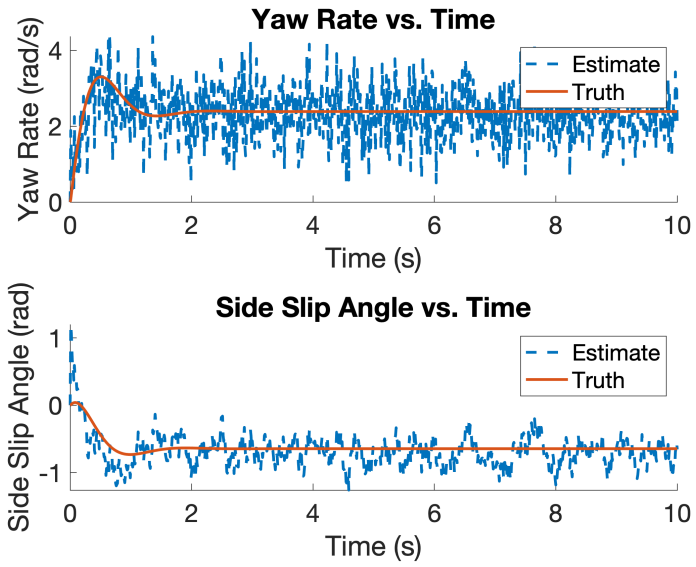

Figure 14: State Estimates & Truth vs. Time

The Q_d calculated to get unbiased state estimates is as follows:

$$Q_d = \begin{bmatrix} 2 & 0 \\ 0 & 0.02 \end{bmatrix}$$

The SNR between the true state and the measurement is ~ -3.9476 while the SNR between the measurement between the esitmate is ~ 2.7786. Clearly there is a fair bit of filtering being done by the Kalman Filter. Compared to the other two filters, the estimate is much noisier, but the trust is more evenly split between the measurement and process model.

Appendix A: Part I Code

```
clear; close all; clc;

currentFile = mfilename('fullpath');
currentFolder = fileparts(currentFile);
addpath(genpath(currentFolder + "/"));

Ts = 0.1;              % Sample Period [s]
fs = 1/Ts;             % Sample Rate [Hz]

```

```
Acl = [ 0    1;
       -1 -1.4];     % Continuous State Transtion
          Matrix
C = [1 0];           % Observation Matrix
Bw = [0 1];          % Input Noise Matrix

sigmaV = 1;          % Sensor Noise Sigma
sigmaQ = 2;          % Process Noise Sigma

% (A)
time = 0:Ts:100;
x_true1 = zeros(2,length(time));
y1 = zeros(2,length(time));
for i = 2:length(time)
    x_true1(:,i) = x_true1(:,i-1) + (Acl*x_true1(:,i
       -1) + Bw*sigmaQ*randn(2,1))*Ts;
    y1(:,i) = C*x_true1(:,i) + sigmaV*randn(2,1);
end

figure('Renderer', 'painters', 'Position', [10 10 900
   600])
hold('on');
title('A) Simulated Measurement vs. Time');
plot(time, y1, 'LineWidth', 2);
xlabel('Time (s)');
ylabel('Measurement');
legend('Position [m]', 'Velocity [m/s]')
ax = gca;
ax.FontSize = 18;

% exportgraphics(gcf, currentFolder + "/../figures/p1a
   .png", 'Resolution', 300);

% (B)
R = sigmaV^2.*eye(size(Acl));    % Measurement
   Covariance Matrix
Q = sigmaQ^2.*eye(size(Acl));    % Continuous Process
   Covariance Matrix
Qd1 = Bw*Q*Bw'*Ts;               % Discrete Process
   Covariance Matrix
Ad = eye(size(Acl)) + Acl.*Ts;   % Discrete State
   Transition Matrix

%% (C)
x1 = 10.*ones(2,length(time));   % State Estimates
L1 = zeros(2,length(time));      % Kalman Gain
```

```
P1 = zeros(2,2,length(time));   % Covariance Matrix
P1(:,:,1) = eye(2);
Pp1 = zeros(2,2,length(time));   % A Priori Covariance
    Matrix
for i = 2:length(time)
    % Time Update
    xp = Ad*x1(:,i-1);
    Pp1(:,:,i) = Ad*P1(:,:,i-1)*Ad' + Qd1;
    % Kalman Gain
    L1(:,i) = (Pp1(:,:,i)*C')/(C*Pp1(:,:,i)*C' +
       sigmaV^2);
    % Measurement Update
    x1(:,i) = xp + L1(:,i)'*(y1(:,i-1) - C*xp);
    P1(:,:,i) = (eye(size(Pp1(:,:,i))) - L1(:,i)*C)*
       Pp1(:,:,i);
end

figure();
tiledlayout(2,1);
nexttile();
hold('on');
plot(time, x1(1,:), '--', 'LineWidth', 2);
plot(time, x_true1(1,:), 'LineWidth', 2);
title('Position vs. Time');
xlabel('Time (s)');
ylabel('Position (m)');
legend('Estimate', 'Truth');
ax = gca;
ax.FontSize = 18;

nexttile();
hold('on');
plot(time, x1(2,:), '--', 'LineWidth', 2);
plot(time, x_true1(2,:), 'LineWidth', 2);
title('Velocity vs. Time');
xlabel('Time (s)');
ylabel('Velocity (m/s)');
legend('Estimate', 'Truth');
ax = gca;
ax.FontSize = 18;

% exportgraphics(gcf, currentFolder + "/../figures/
   p1c1.png", 'Resolution', 300);

figure('Renderer', 'painters', 'Position', [10 10 900
   600])
```

```
89  hold('on');
90  title('C) Kalman Gain vs. Time');
91  plot(time, L1, 'LineWidth', 2);
92  xlabel('Time (s)');
93  ylabel('Kalman Gain');
94  legend('Gain on Position', 'Gain on Velocity');
95  ax = gca;
96  ax.FontSize = 18;
97
98  % exportgraphics(gcf, currentFolder + "/../figures/
        p1c2.png", 'Resolution', 300);
99
100 %% (D)
101 Ppss = squeeze(Pp1(:,:,end));
102 Pss = squeeze(P1(:,:,end));      % Steady State
        Covariance Matrix
103 Lss = L1(:,end);                 % Steady State Kalman
        Gain
104 x2 = 10.*ones(2,length(time));  % State Estimate
105 for i = 2:length(time)
106     % Time Update
107     xp = Ad*x2(:,i-1);
108     % Measurement Update
109     x2(:,i) = xp + Lss'*(y1(:,i-1) - C*xp);
110 end
111
112 poles = eigs(Ad - Lss*C);
113
114 figure();
115 tiledlayout(2,1);
116 nexttile();
117 hold('on');
118 plot(time, x2(1,:), '--', 'LineWidth', 2);
119 plot(time, x_true1(1,:), 'LineWidth', 2);
120 title('Position vs. Time');
121 xlabel('Time (s)');
122 ylabel('Position (m)');
123 legend('Estimate', 'Truth');
124 ax = gca;
125 ax.FontSize = 18;
126
127 nexttile();
128 hold('on');
129 plot(time, x2(2,:), '--', 'LineWidth', 2);
130 plot(time, x_true1(2,:), 'LineWidth', 2);
131 title('Velocity vs. Time');
```

```
xlabel('Time (s)');
ylabel('Velocity (m/s)');
legend('Estimate', 'Truth');
ax = gca;
ax.FontSize = 18;

exportgraphics(gcf, currentFolder + "/../figures/p1d.
    png", 'Resolution', 300);

N1 = norm(std(x_true1 - x1, 0, 2));      % Normal
    Kalman Filter
N2 = norm(std(x_true1 - x2, 0, 2));      % SS Kalman
    Filter

%% (E)
Q_range = 0.1:0.1:10;
R_range = 0.1:0.1:10;
Qidx = 1;
Ridx = 1;
mean_error = zeros(length(Q_range), length(R_range));
for Q = Q_range
    for R = R_range
        Lss = dlqr(Ad, Bw', Q, R);
        x3 = zeros(2,length(time));
        for i = 2:length(time)
            % Time Update
            xp = Ad*x3(:,i-1);
            % Measurement Update
            x3(:,i) = xp + Lss*(y1(:,i-1) - C*xp);
        end
        mean_error(Qidx, Ridx) = norm(mean(x_true1 -
            x3, 2));
        Ridx = Ridx + 1;
    end
    Qidx = Qidx + 1;
    Ridx = 1;
end

minError = min(mean_error, [], "all");
[minQidx, minRidx] = find(mean_error == minError);

minQ = Q_range(minQidx);
minR = R_range(minRidx);
```

Appendix B: Part II Code

```
clear; close all; clc;

currentFile = mfilename('fullpath');
currentFolder = fileparts(currentFile);
addpath(genpath(currentFolder + "/"));

data = readtable("data/hw3_2.txt");
t = data.Var1;
dt = mean(diff(t));
y1 = data.Var2;

R = 1;
Qd1 = 0;

Ac = 0;
Ad = eye(size(Ac)) - Ac*dt;
C = 1;
Bw = 0;

x1 = zeros(1,length(y1));
P1 = 1;
L1 = zeros(1,length(t));
for i = 2:length(t)
    % Time Update
    xp = Ad*x1(:,i-1);
    Pp = Ad*P1*Ad' + Qd1;
    % Kalman Gain
    L1(:,i) = (Pp*C)/(C'*Pp*C + R);
    % Measurement Update
    x1(:,i) = xp + L1(:,i)*(y1(i) - C*xp);
    P1 = (eye(size(Ad)) - L1(:,i)*C)*Pp;
end

figure();
hold('on');
title('Bias & Kalman Gain vs. Time');
plot(t, x1, 'LineWidth', 2);
plot(t, L1, 'LineWidth', 2);
xlabel('Time (s)');
ylabel('Bias & Kalman Gain');
legend('Bias', 'Kalman Gain')
ax = gca;
ax.FontSize = 18;
```

```
44
45 exportgraphics(gcf, currentFolder + "/../figures/p2a.
       png", 'Resolution', 300);
46
47 Qd2 = 0.0001;
48 Qd3 = 0.01;
49
50 P2 = 1;
51 x2 = zeros(1,length(y1));
52 L2 = zeros(1,length(t));
53 P3 = 1;
54 x3 = zeros(1,length(y1));
55 L3 = zeros(1,length(t));
56 for i = 2:length(t)
57     % Time Update
58     xp2 = Ad*x2(:,i-1);
59     Pp2 = Ad*P2*Ad' + Qd2;
60     xp3 = Ad*x3(:,i-1);
61     Pp3 = Ad*P3*Ad' + Qd3;
62     % Kalman Gain
63     L2(:,i) = (Pp2*C)/(C'*Pp2*C + R);
64     L3(:,i) = (Pp3*C)/(C'*Pp3*C + R);
65     % Measurement Update
66     x2(:,i) = xp2 + L2(:,i)*(y1(i) - C*xp2);
67     P2 = (eye(size(Ad)) - L2(:,i)*C)*Pp2;
68     x3(:,i) = xp3 + L3(:,i)*(y1(i) - C*xp3);
69     P3 = (eye(size(Ad)) - L3(:,i)*C)*Pp3;
70 end
71
72 figure();
73 hold('on');
74 title('Bias & Kalman Gain vs. Time');
75 plot(t, x2, 'LineWidth', 2);
76 plot(t, L2, 'LineWidth', 2);
77 plot(t, x3, 'LineWidth', 2);
78 plot(t, L3, 'LineWidth', 2);
79 xlabel('Time (s)');
80 ylabel('Bias & Kalman Gain');
81 legend('Bias; Q_D = 0.0001', 'L_K; Q_D = 0.0001', ...
82        'Bias; Q_D = 0.01', 'L_K; Q_D = 0.01')
83 ax = gca;
84 ax.FontSize = 18;
85
86 exportgraphics(gcf, currentFolder + "/../figures/p2b.
       png", 'Resolution', 300);
87
```

```
% Tradeoff: Settle Time to Noise

numd2 = sqrt(Qd2);
dend2 = [1 (-1+sqrt(Qd2))];
y_filt2 = filter(numd2, dend2, y1, y1(1));
numd3 = sqrt(Qd3);
dend3 = [1 (-1+sqrt(Qd3))];
y_filt3 = filter(numd3, dend3, y1, y1(1));

figure();
hold('on');
title('Bias & Filter vs. Time');
plot(t, x2, 'LineWidth', 2);
plot(t, y_filt2, '--', 'LineWidth', 2);
plot(t, x3, 'LineWidth', 2);
plot(t, y_filt3, '--', 'LineWidth', 2);
xlabel('Time (s)');
ylabel('Bias & Filter');
legend('Bias; Q_D = 0.0001', 'y_{filt}; Q_D = 0.0001',
    ...
       'Bias; Q_D = 0.01', 'y_{filt}; Q_D = 0.01');
ax = gca;
ax.FontSize = 18;

exportgraphics(gcf, currentFolder + "/../figures/p2c.
   png", 'Resolution', 300);
```

Appendix C: Part III Code

```
clear; close all; clc;

currentFile = mfilename('fullpath');
currentFolder = fileparts(currentFile);
addpath(genpath(currentFolder + "/"));
data = readtable("data/hw3_3.txt");

time = data.Var1;
meas_east = data.Var2;
meas_north = data.Var3;
meas_psi = data.Var4;
meas_psi_dot = data.Var5;
meas_vel = data.Var6;

```

```
y = [meas_east, meas_north, meas_psi, meas_psi_dot,
    meas_vel];
dt = mean(diff(time));
N = length(time);
fs = 1/dt;

% (A)
xKF = zeros(5,length(time));
P = eye(5);
Qd = diag([0.01 0.01 0.01 0.001 0.001]);
Rd = diag([0.1 0.1 0.1 0.1 0.1]);

for i = 2:length(time)
    A = zeros(5);
    A(1,3) =  (y(i-1,5) - xKF(4,i-1))*cos(xKF(3,i-1));
    A(1,4) = -sin(xKF(3,i-1));
    A(2,3) = -(y(i-1,5) - xKF(4,i-1))*sin(xKF(3,i-1));
    A(2,4) = -cos(xKF(3,i-1));
    A(3,5) = -1;
    Phi = eye(5) - A*dt;
    % Time Update
    xp = Phi*xKF(:,i-1);
    Pp = Phi*P*Phi' + Qd;
    % Kalman Gain
    H = eye(5);
    H(4,4) = 0;
    H(4,5) = 1;
    H(5,4) = 1;
    H(5,5) = 0;
    L = (Pp*H')/(H*Pp*H' + Rd);
    % Measurement Update
    xKF(:,i) = xp + L*(y(i,:)' - H*xp);
    P = (eye(5) - L*H)*Pp;
end

figure('Renderer', 'painters', 'Position', [10 10 900
    600])
tiledlayout(2,2);
nexttile();
plot(time, xKF(1:2,:), 'LineWidth', 2);
title('East & North vs. Time');
xlabel('Time (s)');
ylabel('Position (m)');
legend('E', 'N');
ax = gca;
ax.FontSize = 18;
```

```
59
60 nexttile();
61 plot(time, rad2deg(xKF(3,:)), 'LineWidth', 2);
62 title('Heading vs. Time');
63 xlabel('Time (s)');
64 ylabel('Heading (deg)');
65 ax = gca;
66 ax.FontSize = 18;
67
68 nexttile([1,2]);
69 plot(time, xKF(4:5,:), 'LineWidth', 2);
70 title('Biases vs. Time');
71 xlabel('Time (s)');
72 ylabel('Bias (m/s | rad/s)');
73 legend('b_r', 'b_g');
74 ax = gca;
75 ax.FontSize = 18;
76
77 exportgraphics(gcf, currentFolder + "/../figures/p3a.
      png", 'Resolution', 300);
78
79 %% (B)
80 batch = 1;
81 xRLS = zeros(5,N);
82 H = eye(5);
83 H(4,4) = 0;
84 H(4,5) = 1;
85 H(5,4) = 1;
86 H(5,5) = 0;
87
88 xRLS(:,1) = pinv(H)*y(1:batch,:)';
89 Qd2 = diag([0.01 0.01 0.01 0.001 0.001]);
90 Rd2 = diag([0.1 0.1 0.1 0.1 0.1]);
91 Qd2(4,4) = 0;
92 Qd2(5,5) = 0;
93 % P = inv(H'*Qd*H);
94 P = eye(5);
95 for i = 2:round((N/batch))
96     start = i*batch; stop = start + batch - 1;
97     K = (P*H')/(H*P*H' + Rd2);
98     P = (eye(5) - K*H)*P;
99     xRLS(:,i) = xRLS(:,i-1) + K*(y(start:stop,:)' - H*
          xRLS(:,i-1));
100 end
101
```

```
figure('Renderer', 'painters', 'Position', [10 10 900 600])
tiledlayout(2,2);
nexttile();
plot(time, xRLS(1:2,:), 'LineWidth', 2);
title('East & North vs. Time');
xlabel('Time (s)');
ylabel('Position (m)');
legend('E', 'N');
ax = gca;
ax.FontSize = 18;

nexttile();
plot(time, rad2deg(xRLS(3,:)), 'LineWidth', 2);
title('Heading vs. Time');
xlabel('Time (s)');
ylabel('Heading (deg)');
ax = gca;
ax.FontSize = 18;

nexttile([1,2]);
plot(time, xRLS(4:5,:), 'LineWidth', 2);
title('Biases vs. Time');
xlabel('Time (s)');
ylabel('Bias (m/s | rad/s)');
legend('b_r', 'b_g');
ax = gca;
ax.FontSize = 18;

exportgraphics(gcf, currentFolder + "/../figures/p3b.png", 'Resolution', 300);

%% (C)
xKF2 = zeros(5,length(time));
P = eye(5);
Qd = diag([0.01 0.01 0.01 0.001 0.001]);
Rd = diag([0.1 0.1 0.1 0.1 0.1]);
for i = 2:length(time)
    A = zeros(5);
    A(1,3) =  (y(i-1,5) - xKF2(4,i-1))*cos(xKF2(3,i-1));
    A(1,4) = -sin(xKF2(3,i-1));
    A(2,3) = -(y(i-1,5) - xKF2(4,i-1))*sin(xKF2(3,i-1));
    A(2,4) = -cos(xKF2(3,i-1));
    A(3,5) = -1;
```

```
144     Phi = eye(5) - A*dt;
145     % Time Update
146     xp = Phi*xKF2(:,i-1);
147     Pp = Phi*P*Phi' + Qd;
148     % Kalman Gain
149     H = eye(5);
150     H(4,4) = 0;
151     H(5,5) = 0;
152     if i < (N - (40/dt))
153         H(4,5) = 1;
154         H(5,4) = 1;
155     end
156     L = (Pp*H')/(H*Pp*H' + Rd);
157     % Measurement Update
158     xKF2(:,i) = xp + L*(y(i,:)' - H*xp);
159     P = (eye(5) - L*H)*Pp;
160 end
161
162 figure('Renderer', 'painters', 'Position', [10 10 900
        600])
163 tiledlayout(2,2);
164 nexttile();
165 plot(time, xKF2(1:2,:), 'LineWidth', 2);
166 title('East & North vs. Time');
167 xlabel('Time (s)');
168 ylabel('Position (m)');
169 legend('E', 'N');
170 ax = gca;
171 ax.FontSize = 18;
172
173 nexttile();
174 plot(time, rad2deg(xKF2(3,:)), 'LineWidth', 2);
175 title('Heading vs. Time');
176 xlabel('Time (s)');
177 ylabel('Heading (deg)');
178 ax = gca;
179 ax.FontSize = 18;
180
181 nexttile([1,2]);
182 plot(time, xKF2(4:5,:), 'LineWidth', 2);
183 title('Biases vs. Time');
184 xlabel('Time (s)');
185 ylabel('Bias (m/s | rad/s)');
186 legend('b_r', 'b_g');
187 ax = gca;
188 ax.FontSize = 18;
```

```
189
190 exportgraphics(gcf, currentFolder + "/../figures/p3c.
       png", 'Resolution', 300);
```

Appendix D: Part IV Code

```
1  %% PART IV
2  clear; close all; clc;
3
4  currentFile = mfilename('fullpath');
5  currentFolder = fileparts(currentFile);
6  addpath(genpath(currentFolder + "/"));
7
8  dt = 0.01;
9  time = 0:dt:10;
10
11 A = [-2.62 12;
12       -0.96 -2];
13 Ad = (eye(size(A)) + A.*dt);
14 B = [14;
15        1];
16 C = [1 0];
17
18 sigmaV = sqrt(0.1);
19
20 R = sigmaV;
21 Q = diag([1 0.03]);
22
23 del = heaviside(time);
24
25 % (A)
26 % Simulation
27 x_true1 = zeros(2,length(time));
28 y1 = zeros(1,length(time));
29 for i = 2:length(time)
30     x_true1(:,i) = x_true1(:, i-1) + (A*x_true1(:, i
          -1) + B*del(i-1))*dt;
31     y1(:,i) = C*x_true1(:,i) + sigmaV*randn;
32 end
33
34 figure();
35 hold('on');
36 plot(time, y1, 'LineWidth', 2);
37 plot(time, x_true1(1,:), 'LineWidth', 2);
```

```
title('Simulated Measurements vs. Time');
xlabel('Time (s)');
ylabel('Yaw-Rate Measurement (rad/s)');
legend('Measurement', 'Truth');
ax = gca;
ax.FontSize = 18;

exportgraphics(gcf, currentFolder + "/../figures/p4a_meas.png", 'Resolution', 300);

% Kalman Filter
x1 = zeros(2,length(time));
P = eye(2);
for i = 2:length(time)
    % Time Update
    xp = Ad*x1(:,i-1);
    Pp = Ad*P*Ad' + Q;
    % Kalman Gain
    L = (Pp*C')/(C*Pp*C' + R);
    % Measurement Update
    x1(:,i) = xp + L*(y1(i) - C*xp);
    P = (eye(2) - L*C)*Pp;
end

mean_error = mean(x_true1 - x1,2);
fprintf('Mean Error of the Yaw Rate Estimate: %0.5g\n', mean_error(1));
fprintf('Mean Error of the Slip Angle Estimate: %0.5g\n\n', mean_error(2));

figure();
t = tiledlayout(2,1);
nexttile();
hold('on');
plot(time, x1(1,:), '--', 'LineWidth', 2);
plot(time, x_true1(1,:), 'LineWidth', 2);
title('Yaw Rate vs. Time');
xlabel('Time (s)');
ylabel('Yaw Rate (rad/s)');
legend('Estimate', 'Truth');
ax = gca;
ax.FontSize = 18;

nexttile();
hold('on');
plot(time, x1(2,:), '--', 'LineWidth', 2);
```

```
plot(time, x_true1(2,:), 'LineWidth', 2);
title('Side Slip Angle vs. Time');
xlabel('Time (s)');
ylabel('Side Slip Angle (rad)');
legend('Estimate', 'Truth');
ax = gca;
ax.FontSize = 18;

exportgraphics(gcf, currentFolder + "/../figures/
    p4a_kf.png", 'Resolution', 300);

%% (B)
A2 = [-2.42  4;
       -0.99 -2];
Ad2 = (eye(2) + A*dt);
B2 = [18
        1];
C2 = C;

Q2 = diag([2 0.06]);

% Simulation
x_true2 = zeros(2,length(time));
y2 = zeros(1,length(time));
for i = 2:length(time)
    x_true2(:,i) = x_true2(:, i-1) + (A*x_true2(:, i
        -1) + B*del(i-1)).*dt;
    y2(:,i) = C*x_true2(:,i) + sigmaV*randn;
end

% Kalman Filter
x2 = zeros(2,length(time));
P = eye(2);
for i = 2:length(time)
    % Time Update
    xp = Ad*x2(:,i-1);
    Pp = Ad*P*Ad' + Q2;
    % Kalman Gain
    L = (Pp*C')/(C*Pp*C' + R);
    % Measurement Update
    x2(:,i) = xp + L*(y2(i) - C*xp);
    P = (eye(2) - L*C)*Pp;
end

mean_error = mean(x_true2 - x2, 2);
```

```
fprintf('Mean Error of the Yaw Rate Estimate: %0.5g\n', mean_error(1));
fprintf('Mean Error of the Slip Angle Estimate: %0.5g\n\n', mean_error(2));

figure();
tiledlayout(2,1);
nexttile();
hold('on');
plot(time, x2(1,:), '--', 'LineWidth', 2);
plot(time, x_true2(1,:), 'LineWidth', 2);
title('Yaw Rate vs. Time');
xlabel('Time (s)');
ylabel('Yaw Rate (rad/s)');
legend('Estimate', 'Truth');
ax = gca;
ax.FontSize = 18;

nexttile();
hold('on');
plot(time, x2(2,:), '--', 'LineWidth', 2);
plot(time, x_true2(2,:), 'LineWidth', 2);
title('Side Slip Angle vs. Time');
xlabel('Time (s)');
ylabel('Side Slip Angle (rad)');
legend('Estimate', 'Truth');
ax = gca;
ax.FontSize = 18;

exportgraphics(gcf, currentFolder + "/../figures/p4b_kf.png", 'Resolution', 300);

%% (C)
sigmaN = sqrt(0.5);
R3 = diag([sigmaV sigmaN]);

% (D)
A3 = A;
B3 = B;
C3 = eye(2);

Q3 = diag([2, 0.02]);

% Simulation
x_true3 = zeros(2,length(time));
y3 = zeros(2,length(time));
```

```
for i = 2:length(time)
    x_true3(:,i) = x_true3(:, i-1) + (A*x_true3(:, i
       -1) + B*del(i-1)).*dt;
    y3(:,i) = C3*x_true3(:,i) + [sigmaV sigmaN]*randn
       (2,1);
end

% Kalman Filter
x3 = zeros(2,length(time));
P = eye(2);
for i = 2:length(time)
    % Time Update
    xp = Ad*x3(:,i-1);
    Pp = Ad*P*Ad' + Q3;
    % Kalman Gain
    L = (Pp*C3')/(C3*Pp*C3' + R3);
    % Measurement Update
    x3(:,i) = xp + L*(y3(:,i) - C3*xp);
    P = (eye(2) - L*C3)*Pp;
end

mean_error = mean(x_true3 - x3, 2);
fprintf('Mean Error of the Yaw Rate Estimate: %0.5g\n'
   , mean_error(1));
fprintf('Mean Error of the Slip Angle Estimate: %0.5g\
   n\n', mean_error(2));

snrratio_meas = snr(x_true3(2,:), y3(2,:));
snratio_estimate = snr(y3(2,:), x3(2,:));

figure();
tiledlayout(2,1);
nexttile();
hold('on');
plot(time, x3(1,:), '--', 'LineWidth', 2);
plot(time, x_true3(1,:), 'LineWidth', 2);
title('Yaw Rate vs. Time');
xlabel('Time (s)');
ylabel('Yaw Rate (rad/s)');
legend('Estimate', 'Truth');
ax = gca;
ax.FontSize = 18;

nexttile();
hold('on');
plot(time, x3(2,:), '--', 'LineWidth', 2);
```

```
plot(time, x_true3(2,:), 'LineWidth', 2);
title('Side Slip Angle vs. Time');
xlabel('Time (s)');
ylabel('Side Slip Angle (rad)');
legend('Estimate', 'Truth');
ax = gca;
ax.FontSize = 18;

exportgraphics(gcf, currentFolder + "/../figures/
   p4d_kf.png", 'Resolution', 300);
```