

# Optimal HW3

Walter Livingston

April 2024

Note: All problems are to be sampled at 10 Hz ( $T_s=0.1$ ) 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:

$$A_{CL} = \begin{bmatrix} 0 & 1 \\ -1 & -1.4 \end{bmatrix} \quad (1)$$

Suppose our measurement is simply position ( $C = [1 \ 0]$ ). 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

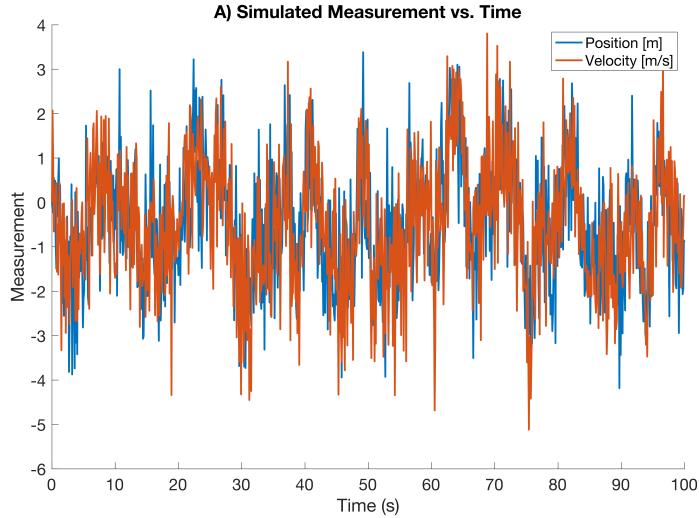


Figure 1: Simulated Measurements vs. Time

## 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

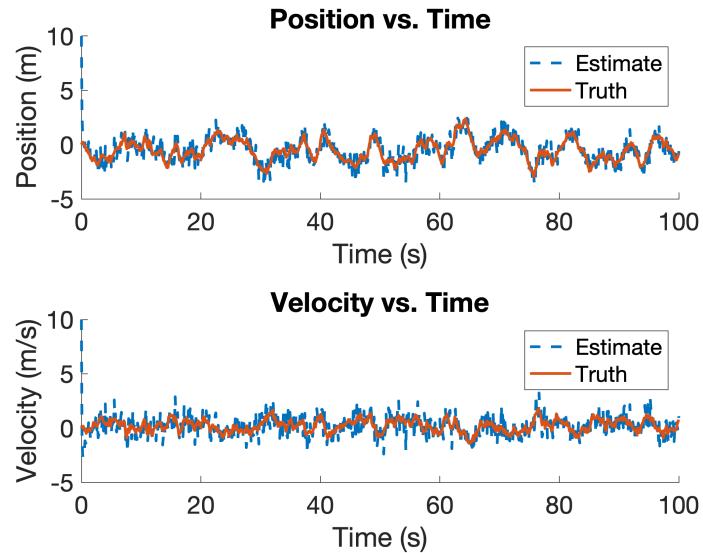


Figure 2: State Estimate vs. Time

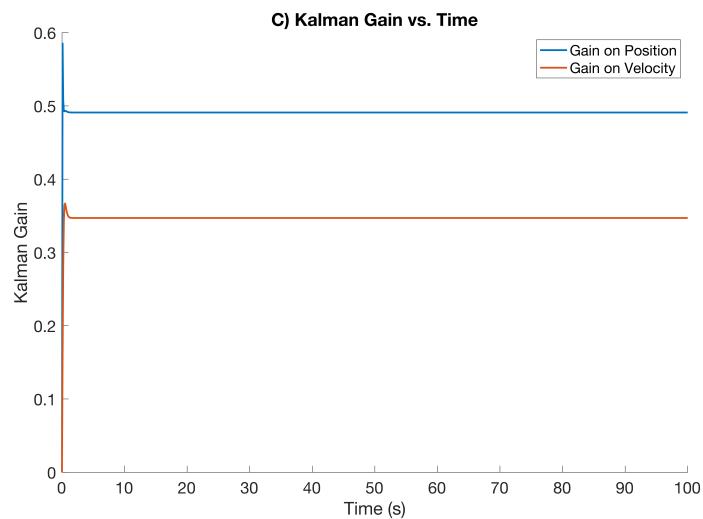


Figure 3: Kalman Gain vs. Time

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

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

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

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

## 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.

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

### Solution

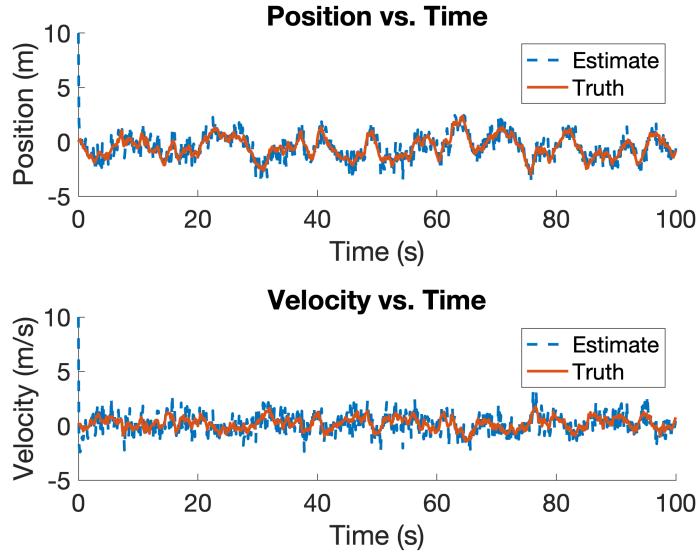


Figure 4: SS Kalman Filter State Estimate vs. Time

$$N = 1.0816$$

$$N_{ss} = 1.0788$$

## 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  $[t \ y]$ . 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:

$$\dot{x} = 0 \quad (3)$$

$$y_k = x_k + \nu_k \quad (4)$$

$$\nu_k \sim N(0, 1) \quad (5)$$

## Part A

Run the Kalman filter estimator with  $Q_d = 0$ . What happens 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

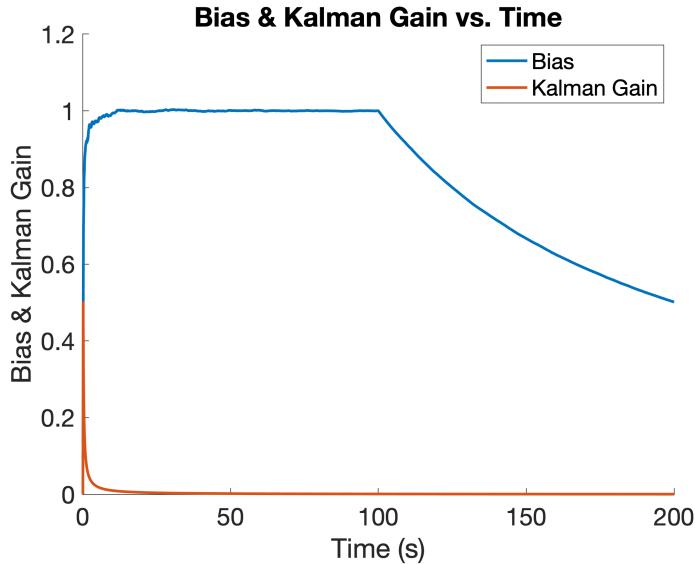


Figure 5: Bias & Kalman Gain vs. Time

At  $t > 100$ , the estimate of the bias slowly starts to decrease. The actual bias has a step change in its 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

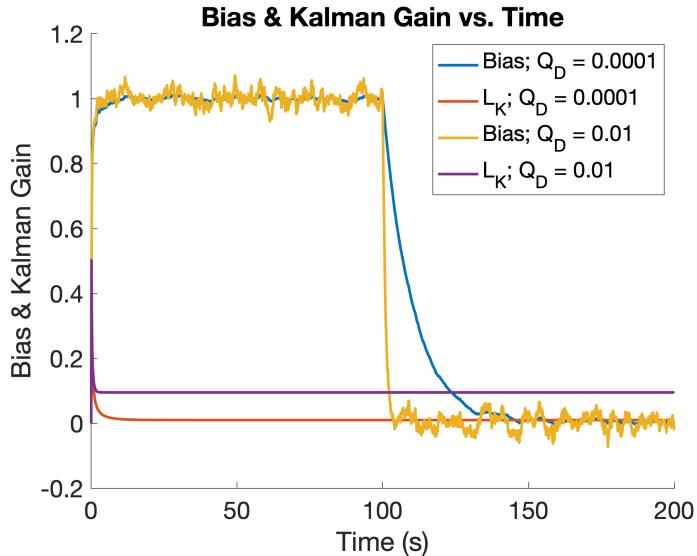


Figure 6: Bias & Kalman Gain vs. Time

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:

$$H(z) = \frac{\text{sqrt}Q_d}{z - (1 - \text{sqrt}Q_d)} \quad (6)$$

use the command: `yf = filter(numd, dend, y, y0)`

## Solution

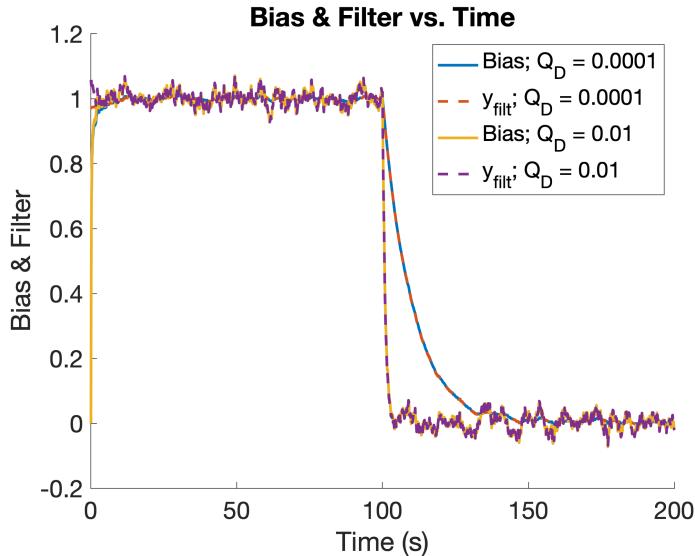


Figure 7: Bias & Filter vs. Time

## 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

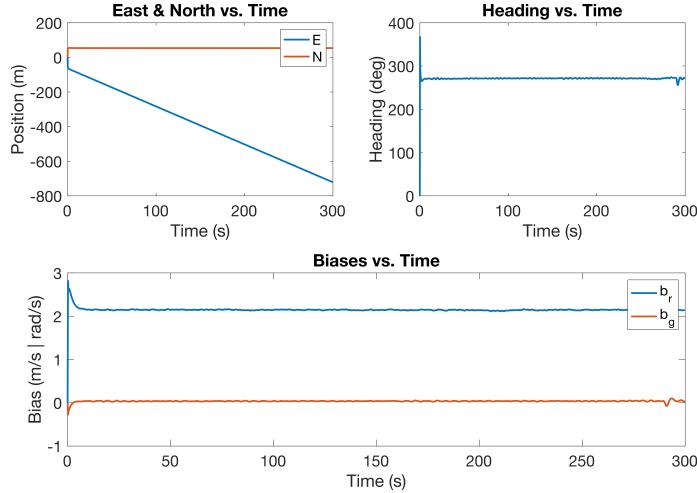


Figure 8: Nav KF States vs. Time

$Q_d$  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

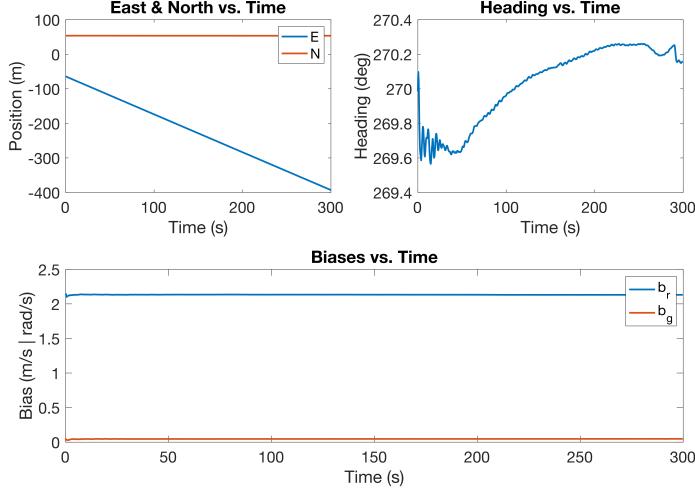


Figure 9: Nav RLS States vs. Time

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

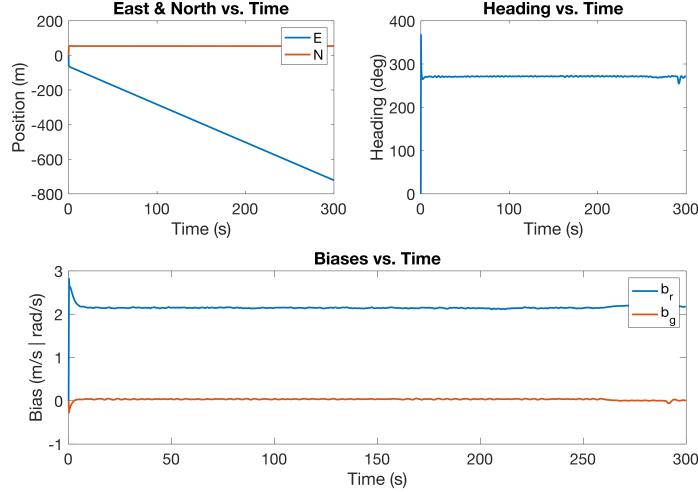


Figure 10: Nav KF States w/ 40s "Outage" vs. Time

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):

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

$$y_k = [1 \ 0] X_k + \nu_k \quad (8)$$

where:

$$\dot{\Psi} - \text{VehicleYawRate} \quad (9)$$

$$\beta - \text{VehicleSideSlipAngle} \quad (10)$$

$$\delta - \text{SteerAngle} \quad (11)$$

$$\nu_k - \text{SampleSensorNoise} \quad (12)$$

### 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.

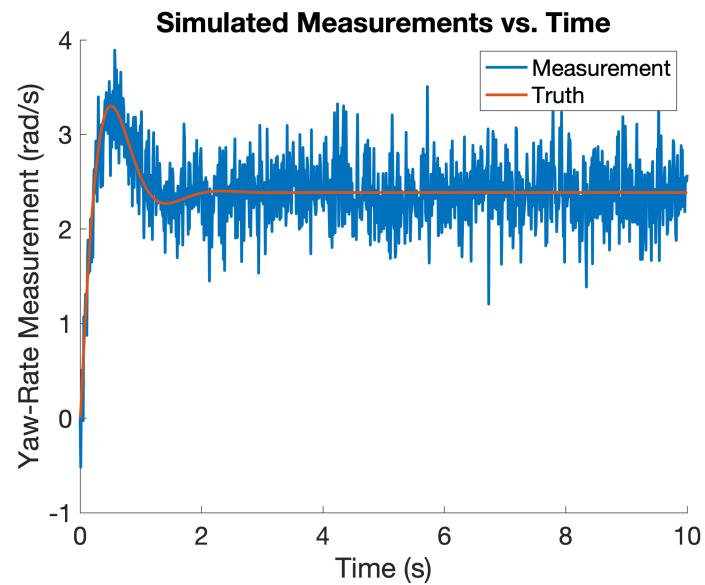


Figure 11: Simulated Measurement vs. Time

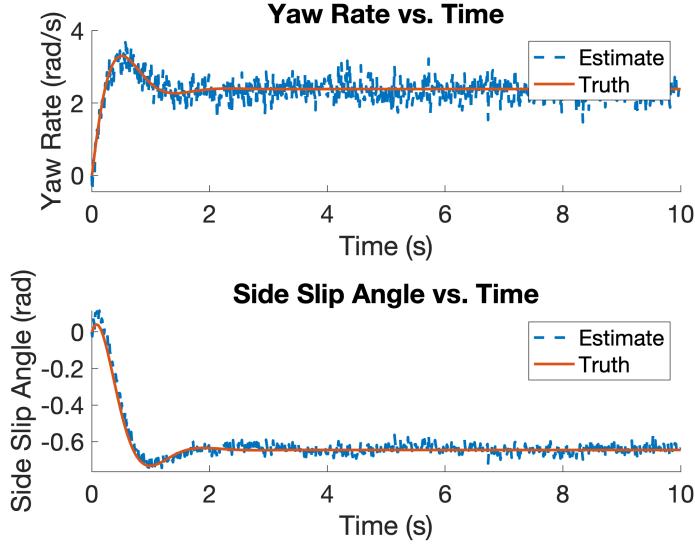


Figure 12: State Estimates & Truth vs. Time

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:

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

$$y_k = [1 \ 0] \vec{X}_k + \nu_k \quad (14)$$

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

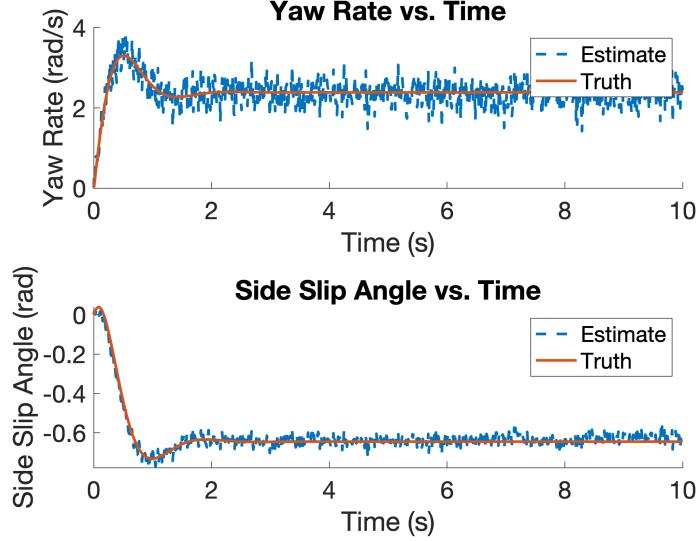


Figure 13: State Estimates & Truth vs. Time

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 ( $\eta_k \sim N(0, 0.5^2)$ ):

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

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

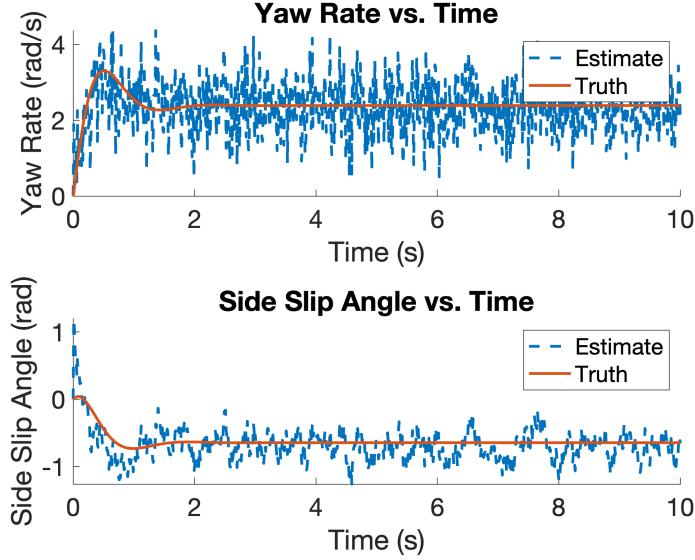


Figure 14: State Estimates & Truth vs. Time

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  $\sim -3.9476$  while the SNR between the measurement and the esitmate is  $\sim 2.7786$ . 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

```

1 clear; close all; clc;
2
3 currentFile = mfilename('fullpath');
4 currentFolder = fileparts(currentFile);
5 addpath(genpath(currentFolder + "/"));
6
7 Ts = 0.1; % Sample Period [s]
8 fs = 1/Ts; % Sample Rate [Hz]
```

```

10 Acl = [ 0      1;
11          -1 -1.4];      % Continuous State Transtion
12          Matrix
13 C = [1 0];           % Observation Matrix
14 Bw = [0 1];          % Input Noise Matrix
15 sigmaV = 1;           % Sensor Noise Sigma
16 sigmaQ = 2;           % Process Noise Sigma
17
18 % (A)
19 time = 0:Ts:100;
20 x_true1 = zeros(2,length(time));
21 y1 = zeros(2,length(time));
22 for i = 2:length(time)
23     x_true1(:,i) = x_true1(:,i-1) + (Acl*x_true1(:,i
24         -1) + Bw*sigmaQ*randn(2,1))*Ts;
25     y1(:,i) = C*x_true1(:,i) + sigmaV*randn(2,1);
26 end
27
28 figure('Renderer', 'painters', 'Position', [10 10 900
29          600])
30 hold('on');
31 title('A) Simulated Measurement vs. Time');
32 plot(time, y1, 'LineWidth', 2);
33 xlabel('Time (s)');
34 ylabel('Measurement');
35 legend('Position [m]', 'Velocity [m/s]')
36 ax = gca;
37 ax.FontSize = 18;
38
39 % exportgraphics(gcf, currentFolder + "/../figures/p1a
40 .png", 'Resolution', 300);
41
42 % (B)
43 R = sigmaV^2.*eye(size(Acl));      % Measurement
44          Covariance Matrix
45 Q = sigmaQ^2.*eye(size(Acl));      % Continuous Process
46          Covariance Matrix
47 Qd1 = Bw*Q*Bw'*Ts;                % Discrete Process
48          Covariance Matrix
49 Ad = eye(size(Acl)) + Acl.*Ts;    % Discrete State
50          Transition Matrix
51
52 %% (C)
53 x1 = 10.*ones(2,length(time));    % State Estimates
54 L1 = zeros(2,length(time));        % Kalman Gain

```

```

48 P1 = zeros(2,2,length(time)); % Covariance Matrix
49 P1(:,:,1) = eye(2);
50 Pp1 = zeros(2,2,length(time)); % A Priori Covariance
   Matrix
51 for i = 2:length(time)
   % Time Update
52     xp = Ad*x1(:,:,i-1);
53     Pp1(:,:,i) = Ad*P1(:,:,i-1)*Ad' + Qd1;
   % Kalman Gain
56     L1(:,:,i) = (Pp1(:,:,i)*C')/(C*Pp1(:,:,i)*C' +
   sigmaV^2);
   % Measurement Update
58     x1(:,:,i) = xp + L1(:,:,i)'*(y1(:,:,i-1) - C*xp);
59     P1(:,:,i) = (eye(size(Pp1(:,:,i)))) - L1(:,:,i)*C*
   Pp1(:,:,i);
60 end
61
62 figure();
63 tiledlayout(2,1);
64 nexttile();
65 hold('on');
66 plot(time, x1(1,:), '--', 'LineWidth', 2);
67 plot(time, x_true1(1,:), 'LineWidth', 2);
68 title('Position vs. Time');
69 xlabel('Time (s)');
70 ylabel('Position (m)');
71 legend('Estimate', 'Truth');
72 ax = gca;
73 ax.FontSize = 18;
74
75 nexttile();
76 hold('on');
77 plot(time, x1(2,:), '--', 'LineWidth', 2);
78 plot(time, x_true1(2,:), 'LineWidth', 2);
79 title('Velocity vs. Time');
80 xlabel('Time (s)');
81 ylabel('Velocity (m/s)');
82 legend('Estimate', 'Truth');
83 ax = gca;
84 ax.FontSize = 18;
85
86 % exportgraphics(gcf, currentFolder + "/../figures/
   pic1.png", 'Resolution', 300);
87
88 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/
99 %             plc2.png', 'Resolution', 300);
100 %% (D)
101 Ppss = squeeze(Pp1(:,:,end));
102 Pss = squeeze(P1(:,:,end)); % Steady State
103 Covariance Matrix
104 Lss = L1(:,end); % Steady State Kalman
105 Gain
106 x2 = 10.*ones(2,length(time)); % State Estimate
107 for i = 2:length(time)
108     % Time Update
109     xp = Ad*x2(:,i-1);
110     % Measurement Update
111     x2(:,i) = xp + Lss'*(y1(:,i-1) - C*xp);
112 end
113
114 poles = eigs(Ad - Lss*C);
115 figure();
116 tiledlayout(2,1);
117 nexttile();
118 hold('on');
119 plot(time, x2(1,:), '--', 'LineWidth', 2);
120 plot(time, x_true1(1,:), 'LineWidth', 2);
121 title('Position vs. Time');
122 xlabel('Time (s)');
123 ylabel('Position (m)');
124 legend('Estimate', 'Truth');
125 ax = gca;
126 ax.FontSize = 18;
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');

```

```

132 xlabel('Time (s)');
133 ylabel('Velocity (m/s)');
134 legend('Estimate', 'Truth');
135 ax = gca;
136 ax.FontSize = 18;
137
138 exportgraphics(gcf, currentFolder + '/../figures/p1d.
    png", 'Resolution', 300);
139
140 N1 = norm(std(x_true1 - x1, 0, 2));           % Normal
    Kalman Filter
141 N2 = norm(std(x_true1 - x2, 0, 2));           % SS Kalman
    Filter
142
143 %% (E)
144 Q_range = 0.1:0.1:10;
145 R_range = 0.1:0.1:10;
146 Qidx = 1;
147 Ridx = 1;
148 mean_error = zeros(length(Q_range), length(R_range));
149 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
165
166 minError = min(mean_error, [], "all");
167 [minQidx, minRidx] = find(mean_error == minError);
168
169 minQ = Q_range(minQidx);
170 minR = R_range(minRidx);

```

## Appendix B: Part II Code

```
1 clear; close all; clc;
2
3 currentFile = mfilename('fullpath');
4 currentFolder = fileparts(currentFile);
5 addpath(genpath(currentFolder + "/"));
6
7 data = readtable("data/hw3_2.txt");
8 t = data.Var1;
9 dt = mean(diff(t));
10 y1 = data.Var2;
11
12 R = 1;
13 Qd1 = 0;
14
15 Ac = 0;
16 Ad = eye(size(Ac)) - Ac*dt;
17 C = 1;
18 Bw = 0;
19
20 x1 = zeros(1,length(y1));
21 P1 = 1;
22 L1 = zeros(1,length(t));
23 for i = 2:length(t)
24     % Time Update
25     xp = Ad*x1(:,i-1);
26     Pp = Ad*P1*Ad' + Qd1;
27     % Kalman Gain
28     L1(:,i) = (Pp*C)/(C'*Pp*C + R);
29     % Measurement Update
30     x1(:,i) = xp + L1(:,i)*(y1(i) - C*xp);
31     P1 = (eye(size(Ad)) - L1(:,i)*C)*Pp;
32 end
33
34 figure();
35 hold('on');
36 title('Bias & Kalman Gain vs. Time');
37 plot(t, x1, 'LineWidth', 2);
38 plot(t, L1, 'LineWidth', 2);
39 xlabel('Time (s)');
40 ylabel('Bias & Kalman Gain');
41 legend('Bias', 'Kalman Gain')
42 ax = gca;
43 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)
    % Time Update
    xp2 = Ad*x2(:,i-1);
    Pp2 = Ad*P2*Ad' + Qd2;
    xp3 = Ad*x3(:,i-1);
    Pp3 = Ad*P3*Ad' + Qd3;
    % Kalman Gain
    L2(:,i) = (Pp2*C)/(C'*Pp2*C + R);
    L3(:,i) = (Pp3*C)/(C'*Pp3*C + R);
    % Measurement Update
    x2(:,i) = xp2 + L2(:,i)*(y1(i) - C*xp2);
    P2 = (eye(size(Ad)) - L2(:,i)*C)*Pp2;
    x3(:,i) = xp3 + L3(:,i)*(y1(i) - C*xp3);
    P3 = (eye(size(Ad)) - L3(:,i)*C)*Pp3;
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', ...
    'Bias; Q_D = 0.01', 'L_K; Q_D = 0.01')
82 ax = gca;
83 ax.FontSize = 18;
84
85
86 exportgraphics(gcf, currentFolder + '/../figures/p2b.
    png", 'Resolution', 300);
87

```

```

88 % Tradeoff: Settle Time to Noise
89
90 numd2 = sqrt(Qd2);
91 dend2 = [1 (-1+sqrt(Qd2))];
92 y_filt2 = filter(numd2, dend2, y1, y1(1));
93 numd3 = sqrt(Qd3);
94 dend3 = [1 (-1+sqrt(Qd3))];
95 y_filt3 = filter(numd3, dend3, y1, y1(1));
96
97 figure();
98 hold('on');
99 title('Bias & Filter vs. Time');
100 plot(t, x2, 'LineWidth', 2);
101 plot(t, y_filt2, '--', 'LineWidth', 2);
102 plot(t, x3, 'LineWidth', 2);
103 plot(t, y_filt3, '--', 'LineWidth', 2);
104 xlabel('Time (s)');
105 ylabel('Bias & Filter');
106 legend('Bias; Q_D = 0.0001', 'y_{filt}; Q_D = 0.0001',
107 ...
108 'Bias; Q_D = 0.01', 'y_{filt}; Q_D = 0.01');
109 ax = gca;
110 ax.FontSize = 18;
111 exportgraphics(gcf, currentFolder + "../figures/p2c.
    png", 'Resolution', 300);

```

## Appendix C: Part III Code

```

1 clear; close all; clc;
2
3 currentFile = mfilename('fullpath');
4 currentFolder = fileparts(currentFile);
5 addpath(genpath(currentFolder + "/"));
6 data = readtable("data/hw3_3.txt");
7
8 time = data.Var1;
9 meas_east = data.Var2;
10 meas_north = data.Var3;
11 meas_psi = data.Var4;
12 meas_psi_dot = data.Var5;
13 meas_vel = data.Var6;
14

```

```

15 y = [meas_east, meas_north, meas_psi, meas_psi_dot,
      meas_vel];
16 dt = mean(diff(time));
17 N = length(time);
18 fs = 1/dt;
19
20 % (A)
21 xKF = zeros(5,length(time));
22 P = eye(5);
23 Qd = diag([0.01 0.01 0.01 0.001 0.001]);
24 Rd = diag([0.1 0.1 0.1 0.1 0.1]);
25
26 for i = 2:length(time)
27     A = zeros(5);
28     A(1,3) = (y(i-1,5) - xKF(4,i-1))*cos(xKF(3,i-1));
29     A(1,4) = -sin(xKF(3,i-1));
30     A(2,3) = -(y(i-1,5) - xKF(4,i-1))*sin(xKF(3,i-1));
31     A(2,4) = -cos(xKF(3,i-1));
32     A(3,5) = -1;
33     Phi = eye(5) - A*dt;
34     % Time Update
35     xp = Phi*xKF(:,i-1);
36     Pp = Phi*P*Phi' + Qd;
37     % Kalman Gain
38     H = eye(5);
39     H(4,4) = 0;
40     H(4,5) = 1;
41     H(5,4) = 1;
42     H(5,5) = 0;
43     L = (Pp*H')/(H*Pp*H' + Rd);
44     % Measurement Update
45     xKF(:,i) = xp + L*(y(i,:)' - H*xp);
46     P = (eye(5) - L*H)*Pp;
47 end
48
49 figure('Renderer', 'painters', 'Position', [10 10 900
600])
50 tiledlayout(2,2);
51 nexttile();
52 plot(time, xKF(1:2,:), 'LineWidth', 2);
53 title('East & North vs. Time');
54 xlabel('Time (s)');
55 ylabel('Position (m)');
56 legend('E', 'N');
57 ax = gca;
58 ax.FontSize = 18;

```

```

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

```

```

102 figure('Renderer', 'painters', 'Position', [10 10 900
103   600])
104 tiledlayout(2,2);
105 nexttile();
106 plot(time, xRLS(1:2,:), 'LineWidth', 2);
107 title('East & North vs. Time');
108 xlabel('Time (s)');
109 ylabel('Position (m)');
110 legend('E', 'N');
111 ax = gca;
112 ax.FontSize = 18;
113
114 nexttile();
115 plot(time, rad2deg(xRLS(3,:)), 'LineWidth', 2);
116 title('Heading vs. Time');
117 xlabel('Time (s)');
118 ylabel('Heading (deg)');
119 ax = gca;
120 ax.FontSize = 18;
121
122 nexttile([1,2]);
123 plot(time, xRLS(4:5,:), 'LineWidth', 2);
124 title('Biases vs. Time');
125 xlabel('Time (s)');
126 ylabel('Bias (m/s | rad/s)');
127 legend('b_r', 'b_g');
128 ax = gca;
129 ax.FontSize = 18;
130
131 exportgraphics(gcf, currentFolder + '/../figures/p3b.
132   png", 'Resolution', 300);
133
134 %% (C)
135 xKF2 = zeros(5,length(time));
136 P = eye(5);
137 Qd = diag([0.01 0.01 0.01 0.001 0.001]);
138 Rd = diag([0.1 0.1 0.1 0.1 0.1]);
139 for i = 2:length(time)
140   A = zeros(5);
141   A(1,3) = (y(i-1,5) - xKF2(4,i-1))*cos(xKF2(3,i-1)
142     );
143   A(1,4) = -sin(xKF2(3,i-1));
144   A(2,3) = -(y(i-1,5) - xKF2(4,i-1))*sin(xKF2(3,i-1)
145     );
146   A(2,4) = -cos(xKF2(3,i-1));
147   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
163 600])
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);

```

```

38 title('Simulated Measurements vs. Time');
39 xlabel('Time (s)');
40 ylabel('Yaw-Rate Measurement (rad/s)');
41 legend('Measurement', 'Truth');
42 ax = gca;
43 ax.FontSize = 18;
44
45 exportgraphics(gcf, currentFolder + '/../figures/
    p4a_meas.png", 'Resolution', 300);
46
47 % Kalman Filter
48 x1 = zeros(2,length(time));
49 P = eye(2);
50 for i = 2:length(time)
51     % Time Update
52     xp = Ad*x1(:,i-1);
53     Pp = Ad*P*Ad' + Q;
54     % Kalman Gain
55     L = (Pp*C')/(C*Pp*C' + R);
56     % Measurement Update
57     x1(:,i) = xp + L*(y1(i) - C*xp);
58     P = (eye(2) - L*C)*Pp;
59 end
60
61 mean_error = mean(x_true1 - x1,2);
62 fprintf('Mean Error of the Yaw Rate Estimate: %0.5g\n'
    , mean_error(1));
63 fprintf('Mean Error of the Slip Angle Estimate: %0.5g\
    \n\n', mean_error(2));
64
65 figure();
66 t = tiledlayout(2,1);
67 nexttile();
68 hold('on');
69 plot(time, x1(1,:), '--', 'LineWidth', 2);
70 plot(time, x_true1(1,:), 'LineWidth', 2);
71 title('Yaw Rate vs. Time');
72 xlabel('Time (s)');
73 ylabel('Yaw Rate (rad/s)');
74 legend('Estimate', 'Truth');
75 ax = gca;
76 ax.FontSize = 18;
77
78 nexttile();
79 hold('on');
80 plot(time, x1(2,:), '--', 'LineWidth', 2);

```

```

81 plot(time, x_true1(2,:), 'LineWidth', 2);
82 title('Side Slip Angle vs. Time');
83 xlabel('Time (s)');
84 ylabel('Side Slip Angle (rad)');
85 legend('Estimate', 'Truth');
86 ax = gca;
87 ax.FontSize = 18;
88
89 exportgraphics(gcf, currentFolder + '/../figures/
    p4a_kf.png', 'Resolution', 300);
90
91 %% (B)
92 A2 = [-2.42 4;
93         -0.99 -2];
94 Ad2 = (eye(2) + A*dt);
95 B2 = [18
96         1];
97 C2 = C;
98
99 Q2 = diag([2 0.06]);
100
101 % Simulation
102 x_true2 = zeros(2,length(time));
103 y2 = zeros(1,length(time));
104 for i = 2:length(time)
105     x_true2(:,i) = x_true2(:, i-1) + (A*x_true2(:, i-1) + B*del(i-1)).*dt;
106     y2(:,i) = C*x_true2(:,i) + sigmaV*randn;
107 end
108
109 % Kalman Filter
110 x2 = zeros(2,length(time));
111 P = eye(2);
112 for i = 2:length(time)
113     % Time Update
114     xp = Ad*x2(:,i-1);
115     Pp = Ad*P*Ad' + Q2;
116     % Kalman Gain
117     L = (Pp*C')/(C*Pp*C' + R);
118     % Measurement Update
119     x2(:,i) = xp + L*(y2(i) - C*xp);
120     P = (eye(2) - L*C)*Pp;
121 end
122
123 mean_error = mean(x_true2 - x2, 2);

```

```

124 fprintf('Mean Error of the Yaw Rate Estimate: %0.5g\n'
125     , mean_error(1));
126 fprintf('Mean Error of the Slip Angle Estimate: %0.5g\
127 \n', mean_error(2));
128
129 figure();
130 tiledlayout(2,1);
131 nexttile();
132 hold('on');
133 plot(time, x2(1,:), '--', 'LineWidth', 2);
134 plot(time, x_true2(1,:), 'LineWidth', 2);
135 title('Yaw Rate vs. Time');
136 xlabel('Time (s)');
137 ylabel('Yaw Rate (rad/s)');
138 legend('Estimate', 'Truth');
139 ax = gca;
140 ax.FontSize = 18;
141
142 nexttile();
143 hold('on');
144 plot(time, x2(2,:), '--', 'LineWidth', 2);
145 plot(time, x_true2(2,:), 'LineWidth', 2);
146 title('Side Slip Angle vs. Time');
147 xlabel('Time (s)');
148 ylabel('Side Slip Angle (rad)');
149 legend('Estimate', 'Truth');
150 ax = gca;
151 ax.FontSize = 18;
152
153 %% (C)
154 sigmaN = sqrt(0.5);
155 R3 = diag([sigmaV sigmaN]);
156
157 %% (D)
158 A3 = A;
159 B3 = B;
160 C3 = eye(2);
161
162 Q3 = diag([2, 0.02]);
163
164 %% Simulation
165 x_true3 = zeros(2,length(time));
166 y3 = zeros(2,length(time));

```

```

167 for i = 2:length(time)
168     x_true3(:,i) = x_true3(:, i-1) + (A*x_true3(:, i
169         -1) + B*del(i-1)).*dt;
170     y3(:,i) = C3*x_true3(:,i) + [sigmaV sigmaN]*randn
171         (2,1);
172 end
173 % Kalman Filter
174 x3 = zeros(2,length(time));
175 P = eye(2);
176 for i = 2:length(time)
177     % Time Update
178     xp = Ad*x3(:,i-1);
179     Pp = Ad*P*Ad' + Q3;
180     % Kalman Gain
181     L = (Pp*C3')/(C3*Pp*C3' + R3);
182     % Measurement Update
183     x3(:,i) = xp + L*(y3(:,i) - C3*xp);
184     P = (eye(2) - L*C3)*Pp;
185 end
186 mean_error = mean(x_true3 - x3, 2);
187 fprintf('Mean Error of the Yaw Rate Estimate: %0.5g\n'
188     , mean_error(1));
189 fprintf('Mean Error of the Slip Angle Estimate: %0.5g\
190     \n\n', mean_error(2));
191 snrratio_meas = snr(x_true3(2,:), y3(2,:));
192 snratio_estimate = snr(y3(2,:), x3(2,:));
193 figure();
194 tiledlayout(2,1);
195 nexttile();
196 hold('on');
197 plot(time, x3(1,:), '--', 'LineWidth', 2);
198 plot(time, x_true3(1,:), 'LineWidth', 2);
199 title('Yaw Rate vs. Time');
200 xlabel('Time (s)');
201 ylabel('Yaw Rate (rad/s)');
202 legend('Estimate', 'Truth');
203 ax = gca;
204 ax.FontSize = 18;
205 nexttile();
206 hold('on');
207 plot(time, x3(2,:), '--', 'LineWidth', 2);

```

```
209 plot(time, x_true3(2,:), 'LineWidth', 2);
210 title('Side Slip Angle vs. Time');
211 xlabel('Time (s)');
212 ylabel('Side Slip Angle (rad)');
213 legend('Estimate', 'Truth');
214 ax = gca;
215 ax.FontSize = 18;
216
217 exportgraphics(gcf, currentFolder + '/../figures/
    p4d_kf.png', 'Resolution', 300);
```