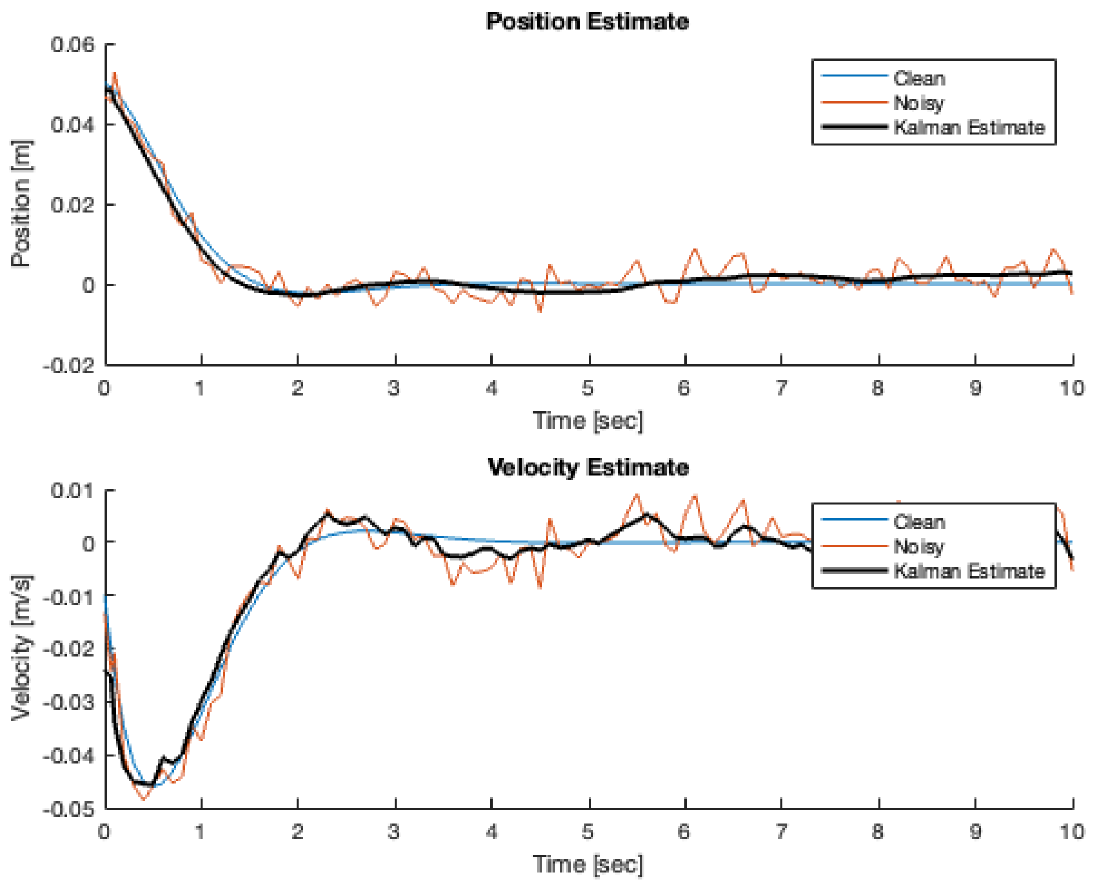
Gagandeep Thapar; HW6 AERO 560

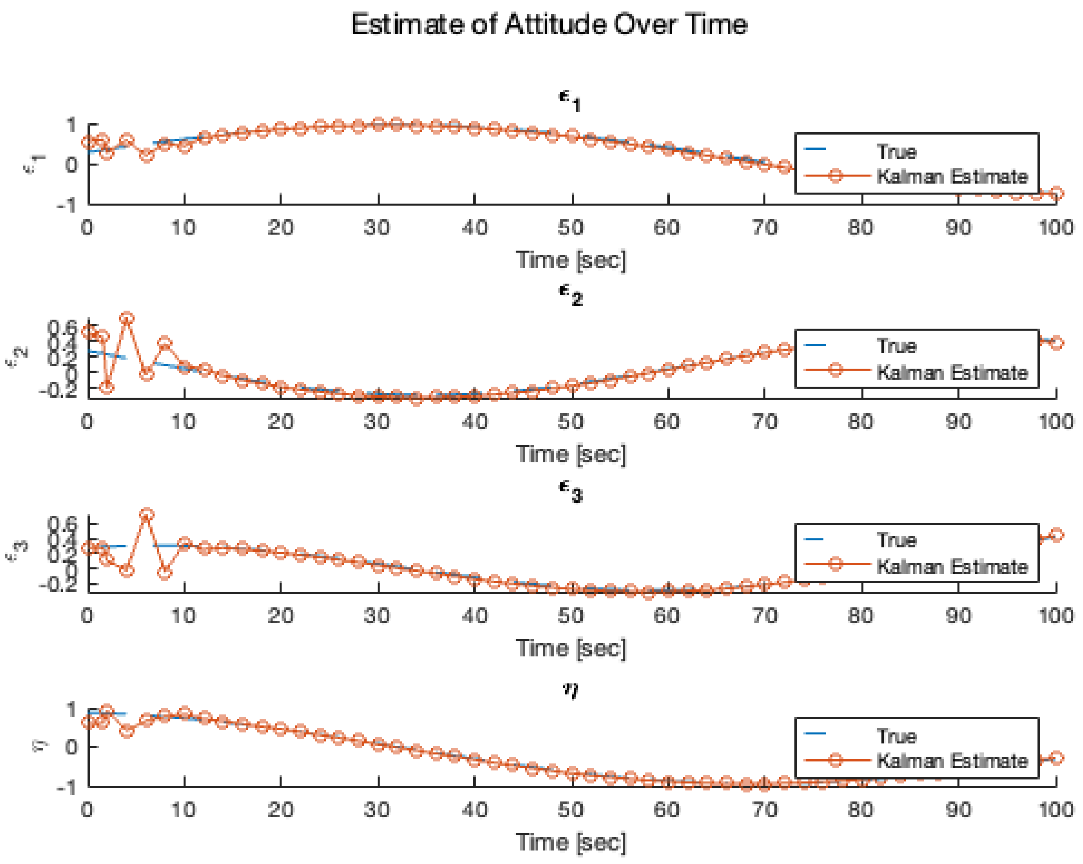
Table of Contents

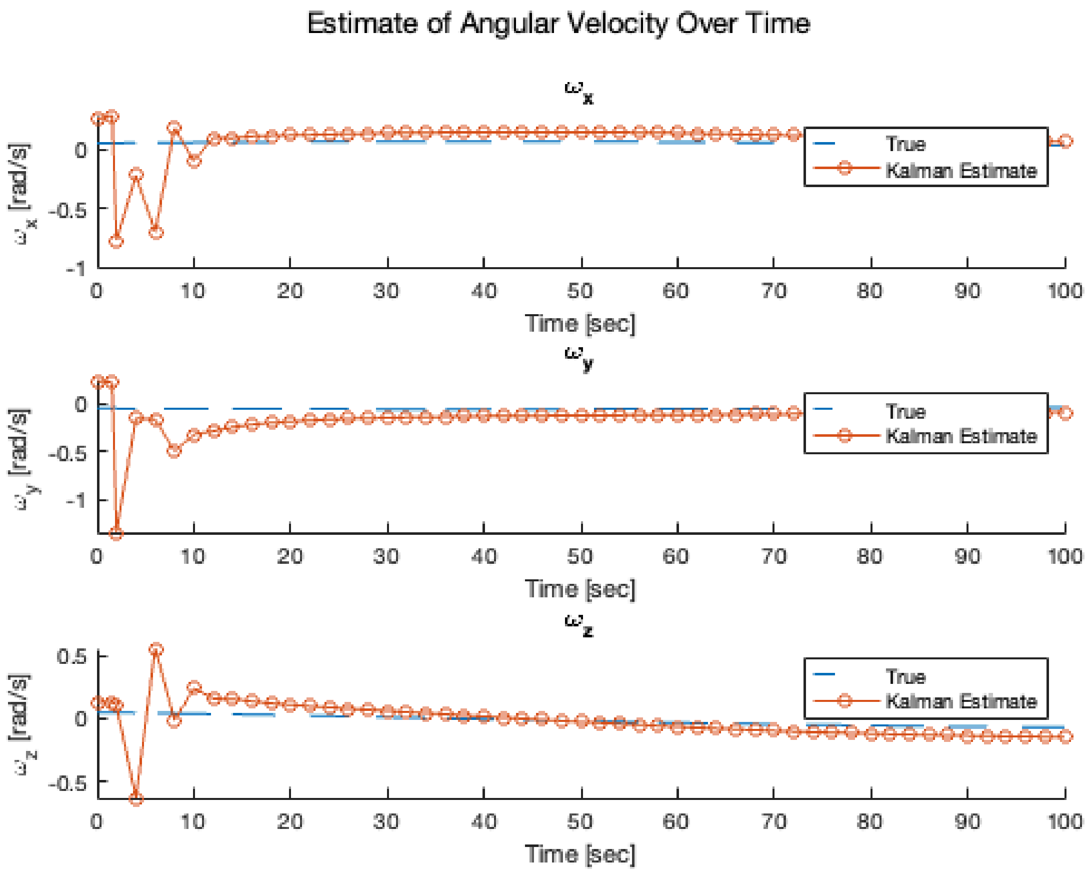
Housekeeping

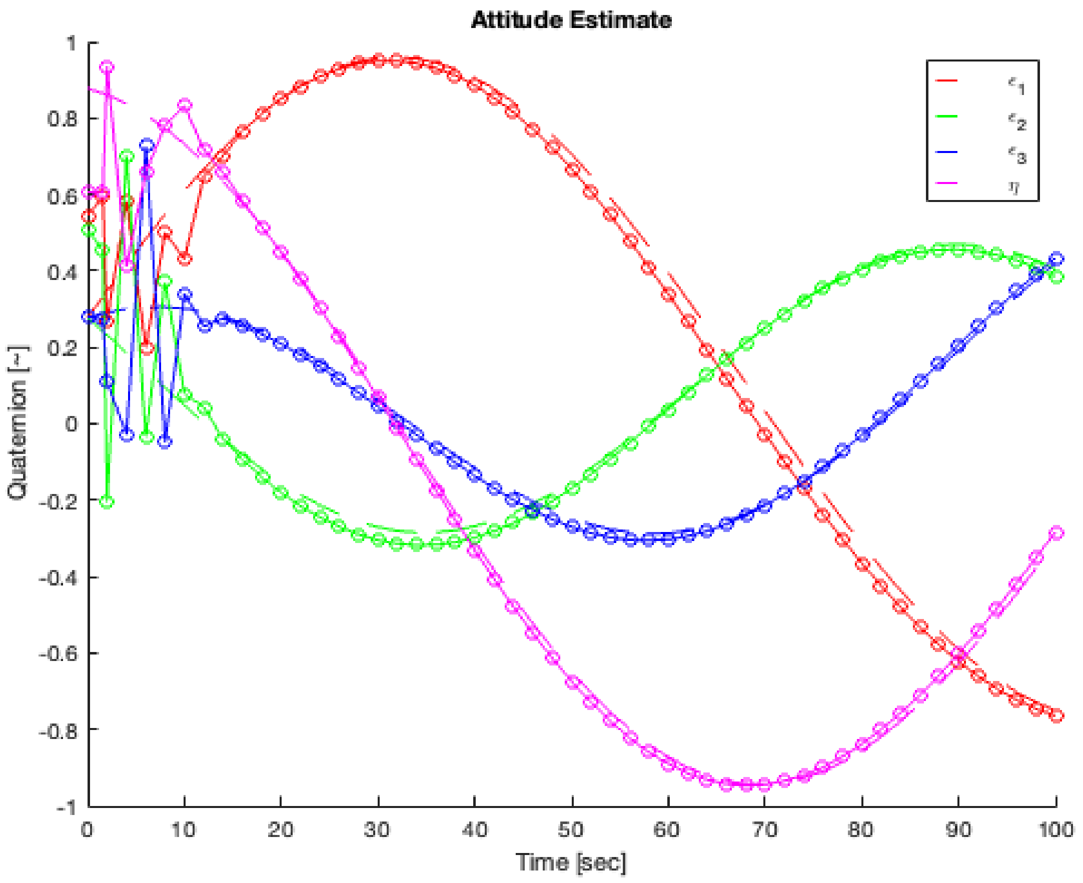
Part 1

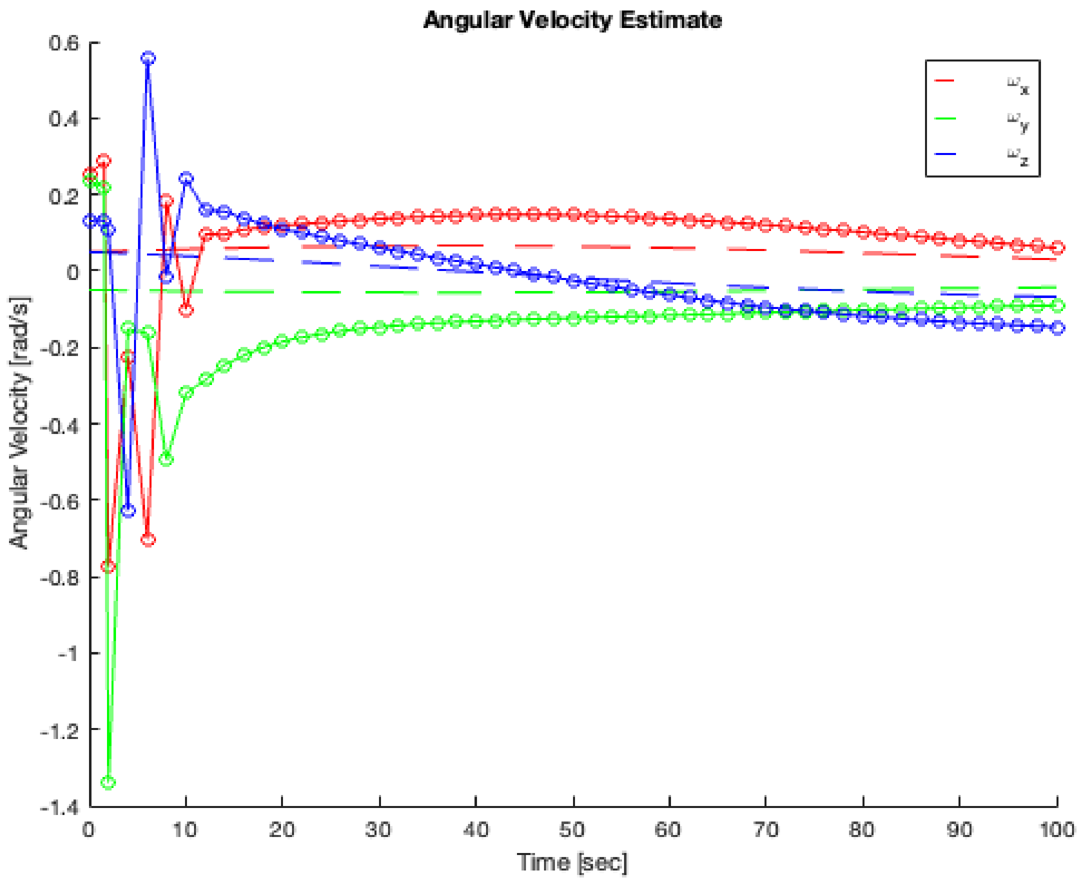


Part 2


Estimate of Attitude Over Time
ϵ_1
ϵ_2
ϵ_3
η
Time [sec]
True
Kalman Estimate


Estimate of Angular Velocity Over Time
ω'x
ω'x [rad/s]
Time [sec]
True
Kalman Estimate
ω'y
ω'y [rad/s]
ω'z
ω'z [rad/s]


Attitude Estimate
Quaternion [~]
Time [sec]
ϵ_1
ϵ_2
ϵ_3
η
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
10
20
30
40
50
60
70
80
90
100



Published with MATLAB® R2022b

Gagandeep Thapar; HW6 AERO 560

Table of Contents

Housekeeping

```
clc;
clear all;
close all;

% random seed
seed = floor(sum(clock));
```

Part 1

```
% givens
p1.wn = 2;
p1.zeta = 0.7;
p1.delT = 0.1;

p1.x0 = 0.05;
p1.xd0 = -0.01;
p1.Q = 0.1;
p1.R = 0.1;
p1.sigma = 0.0035;
p1.u = 0;

% work
p1.A = [0 1; -p1.wn^2 -2*p1.zeta*p1.wn];
p1.B = [0;p1.wn^2];
p1.C = eye(2);
p1.D = zeros(2,1);
p1.state0 = [p1.x0;p1.xd0];

p1.sysC = ss(p1.A, p1.B, p1.C, p1.D);
p1.sysD = c2d(p1.sysC, p1.delT);

p1.F = p1.sysD.A;
p1.G = p1.sysD.B;
p1.H = p1.sysD.C;

p1.L = p1.G;
p1.M = p1.H;

% simulate
```

```
p1.outC = sim("AERO560_HW6Sim_Thapar.slx");
p1.t = squeeze(p1.outC.tout)';
p1.yC = squeeze(p1.outC.y_Clean)';
p1.yN = squeeze(p1.outC.y_Noisy)';
p1.xHat = squeeze(p1.outC.xHat);

% plot
figure
subplot(2,1,1)
hold on
plot(p1.t, p1.yC(:,1))
plot(p1.t, p1.yN(:,1))
plot(p1.t, p1.xHat(:,1),'k', 'LineWidth', 2)
hold off
legend('Clean', 'Noisy', 'Kalman Estimate')
title('Position Estimate')
xlabel('Time [sec]')
ylabel('Position [m]')

subplot(2,1,2)
hold on
plot(p1.t, p1.yC(:,2))
plot(p1.t, p1.yN(:,2))
plot(p1.t, p1.xHat(:,2),'k', 'LineWidth', 2)
hold off
legend('Clean', 'Noisy', 'Kalman Estimate')
title('Velocity Estimate')
xlabel('Time [sec]')
ylabel('Velocity [m/s]')
```

Part 2

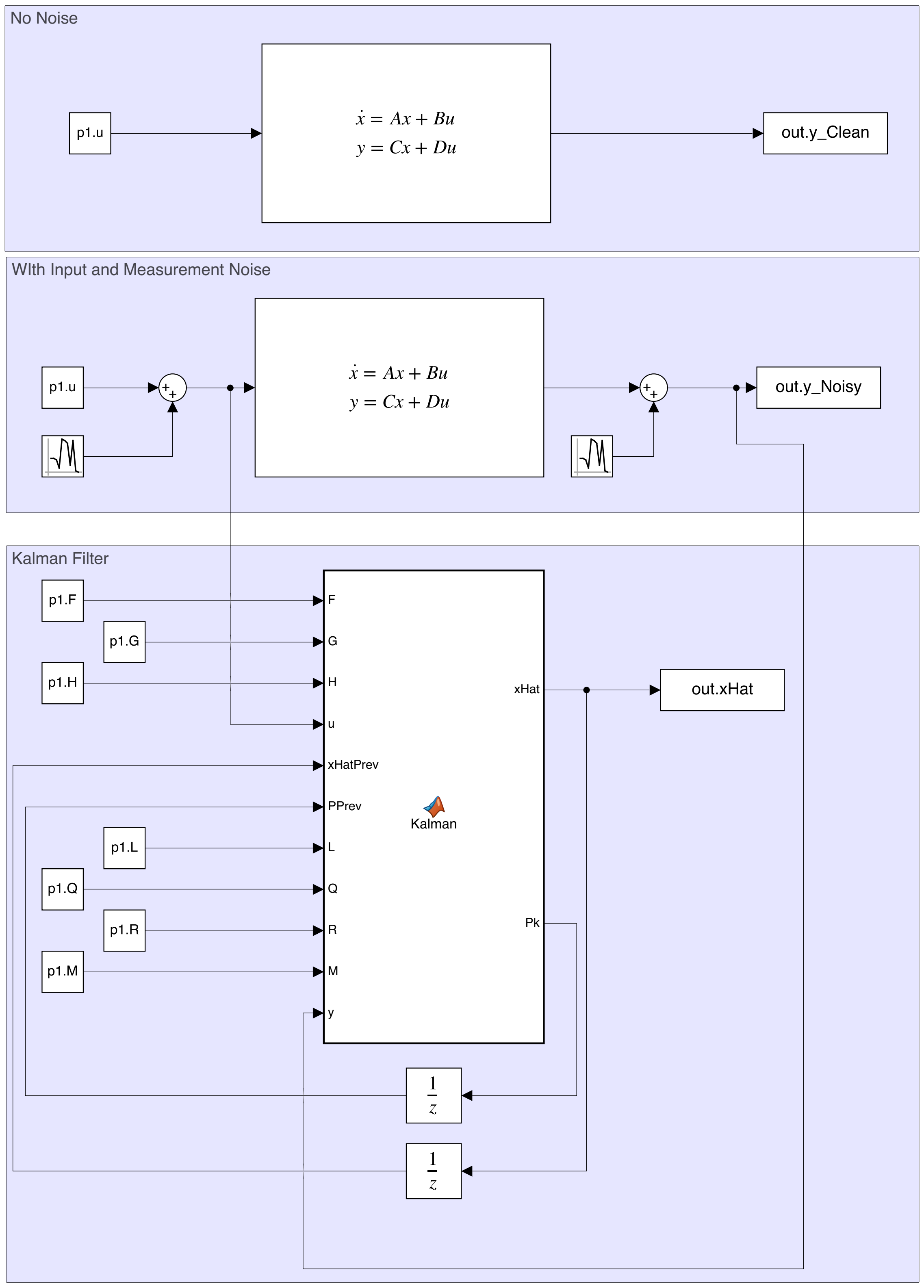
```
% givens
p2.delT = 1;
p2.sA = [1;0;0];
p2.bA = [0.3815;-0.0969;0.9193];

p2.u = 0;
p2.J = diag([27, 17, 25]);

p2.w0 = [0.05;-0.05;0.05];
p2.e0 = sin(0.5)/sqrt(3) * [1;1;1];
p2.n0 = cos(0.5);

p2.wHat0 = zeros(3,1);
p2.eHat0 = zeros(3,1);
p2.nHat0 = 1;

p2.Pk0 = 0.1*eye(7);

p2.sigQ = 0.001;
p2.Q = p2.sigQ^2;
```

```
p2.sigM = 0.01;
p2.sigS = 0.005;
p2.R = diag([p2.sigM^2, p2.sigM^2, p2.sigM^2, p2.sigS^2, p2.sigS^2,
 p2.sigS^2]);

p2.v = 0;

% simulate
p2.out = sim('AERO560_HW6_Sim2_Thapar.slx', 100);

p2.t = squeeze(p2.out.tout);
p2.wTrue = squeeze(p2.out.wTrue);
p2.eTrue = squeeze(p2.out.eTrue);
p2.nTrue = squeeze(p2.out.nTrue)';
p2.xHat = squeeze(p2.out.xHat);

p2.wHat = p2.xHat(:,1:3);
p2.eHat = p2.xHat(:,4:6);
p2.nHat = p2.xHat(:,7);

% plot
figure
subplot(4,1,1)
hold on
plot(p2.t, p2.eTrue(:,1), '--')
plot(p2.t, p2.eHat(:,1), '-o')
hold off
xlabel('Time [sec]')
ylabel('\epsilon_1')
title('\epsilon_1')
legend('True', 'Kalman Estimate')

subplot(4,1,2)
hold on
plot(p2.t, p2.eTrue(:,2),'--')
plot(p2.t, p2.eHat(:,2), '-o')
hold off
xlabel('Time [sec]')
ylabel('\epsilon_2')
title('\epsilon_2')
legend('True', 'Kalman Estimate')

subplot(4,1,3)
hold on
plot(p2.t, p2.eTrue(:,3),'--')
plot(p2.t, p2.eHat(:,3), '-o')
hold off
xlabel('Time [sec]')
ylabel('\epsilon_3')
title('\epsilon_3')
legend('True', 'Kalman Estimate')

subplot(4,1,4)
hold on
```

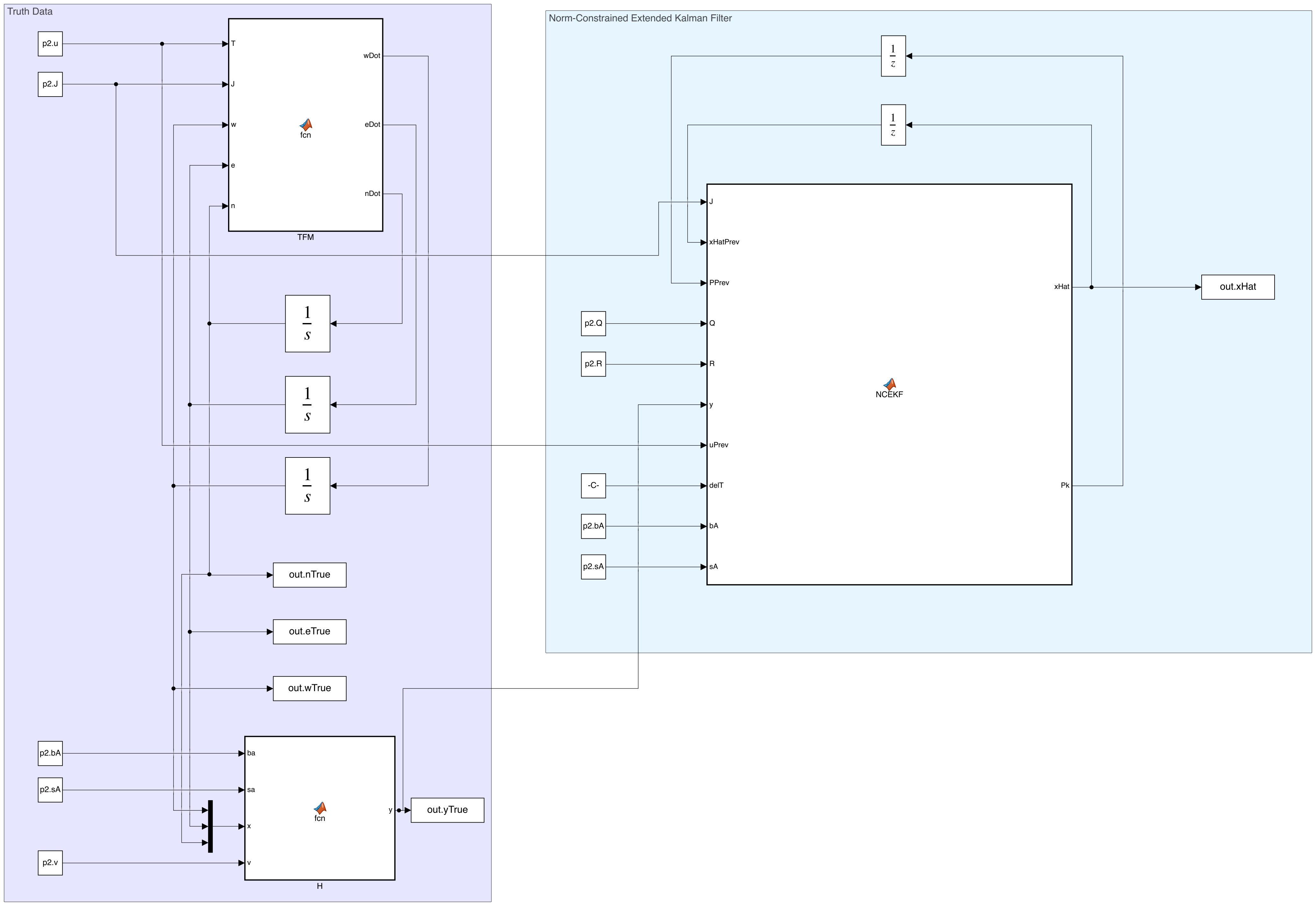
```
plot(p2.t, p2.nTrue, '--')
plot(p2.t, p2.nHat, '-o')
hold off
xlabel('Time [sec]')
ylabel('\eta')
title('\eta')
legend('True', 'Kalman Estimate')
sgtitle('Estimate of Attitude Over Time')

figure
subplot(3,1,1)
hold on
plot(p2.t, p2.wTrue(:,1), '--')
plot(p2.t, p2.wHat(:,1), '-o')
hold off
xlabel('Time [sec]')
ylabel('\omega_x [rad/s]')
title('\omega_x')
legend('True', 'Kalman Estimate')

subplot(3,1,2)
hold on
plot(p2.t, p2.wTrue(:,2), '--')
plot(p2.t, p2.wHat(:,2), '-o')
hold off
xlabel('Time [sec]')
ylabel('\omega_y [rad/s]')
title('\omega_y')
legend('True', 'Kalman Estimate')

subplot(3,1,3)
hold on
plot(p2.t, p2.wTrue(:,3), '--')
plot(p2.t, p2.wHat(:,3), '-o')
hold off
xlabel('Time [sec]')
ylabel('\omega_z [rad/s]')
title('\omega_z')
legend('True', 'Kalman Estimate')
sgtitle('Estimate of Angular Velocity Over Time')

figure
hold on
plot(p2.t, p2.eTrue(:,1), '--r')
plot(p2.t, p2.eTrue(:,2), '--g')
plot(p2.t, p2.eTrue(:,3), '--b')
plot(p2.t, p2.nTrue, '--m')

plot(p2.t, p2.eHat(:,1), '-or');
plot(p2.t, p2.eHat(:,2), '-og');
plot(p2.t, p2.eHat(:,3), '-ob');
plot(p2.t, p2.nHat, '-om');

hold off
```

```
legend('\epsilon_1', '\epsilon_2', '\epsilon_3', '\eta')
xlabel('Time [sec]')
ylabel('Quaternion [~]')
title('Attitude Estimate')

figure
hold on
plot(p2.t, p2.wTrue(:,1), '--r')
plot(p2.t, p2.wTrue(:,2), '--g')
plot(p2.t, p2.wTrue(:,3), '--b')

plot(p2.t, p2.wHat(:,1), '-or')
plot(p2.t, p2.wHat(:,2), '-og')
plot(p2.t, p2.wHat(:,3), '-ob')

hold off
legend('\omega_x', '\omega_y', '\omega_z')
xlabel('Time [sec]')
ylabel('Angular Velocity [rad/s]')
title('Angular Velocity Estimate')
```

Published with MATLAB® R2022b


No Noise
p1.u
$\dot{x} = Ax + Bu$
$y = Cx + Du$
out.y_Clean
WIth Input and Measurement Noise
p1.u
$\dot{x} = Ax + Bu$
$y = Cx + Du$
out.y_Noisy
Kalman Filter
p1.F
p1.G
p1.H
p1.L
p1.Q
p1.R
p1.M
F
G
H
u
xHatPrev
PPrev
L
Q
R
M
y
Kalman
xHat
Pk
out.xHat
$\frac{1}{z}$
$\frac{1}{z}$

```
function [xHat, Pk] = Kalman(F, G, H, u, xHatPrev, PPrev, L, Q, R, M, y)

% predictor
xHatMinus = F*xHatPrev + G*u;
PkMinus = F*PPrev*F' + L*Q*L';

% corrector
W = H*PkMinus*H' + M*R*M';
K = PkMinus * H' * inv(W);

yHatMinus = H*xHatMinus;

xHat = xHatMinus + K*(y - yHatMinus);
Pk = PkMinus - K*H*PkMinus - PkMinus*H'*K' + K*W*K';

end
```


Truth Data
p2.u
p2.J
T
J
w
e
n
fcn
wDot
eDot
nDot
TFM
1/s
1/s
1/s
out.nTrue
out.eTrue
out.wTrue
p2.bA
p2.sA
p2.v
ba
sa
x
v
fcn
y
H
out.yTrue
Norm-Constrained Extended Kalman Filter
1/z
1/z
J
xHatPrev
PPrev
Q
R
y
uPrev
delT
bA
sA
p2.Q
p2.R
-C-
p2.bA
p2.sA
NCEKF
xHat
Pk
out.xHat

```
function y = fcn(ba, sa, x, v)

    function wx = skewSymmetric(w)
        wx = [0, -1*w(3), w(2);
             w(3), 0, -1*w(1);
             -1*w(2), w(1), 0];
    end


    function Yaq = Y_aa_q(a, e, n)

        Ya = [n*eye(3)-skewSymmetric(e), -e];
        Yb = [skewSymmetric(a), a; -a', 0];
        Yaq = Ya*Yb;

    end

e = x(4:6);
n = x(7);

y = [zeros(3,3), Y_aa_q(ba, e, n);
     zeros(3,3), Y_aa_q(sa, e, n)] * x + v;

end
```

```
function [xHat, Pk] = NCEKF(J, xHatPrev, PPrev, Q, R, y, uPrev, delT, bA, sA)

    function wx = skewSymmetric(w)
        wx = [0, -1*w(3), w(2);
             w(3), 0, -1*w(1);
             -1*w(2), w(1), 0];
    end

    function Yaq = Y_aa_q(a, e, n)

        Ya = [n*eye(3)-skewSymmetric(e), -e];
        Yb = [skewSymmetric(a), a; -a', 0];
        Yaq = Ya*Yb;

    end

    function dstate = TFM(state, T, J)

        w = state(1:3);
        e = state(4:6);
        n = state(7);

        wDot = J\(T-skewSymmetric(w)*J*w);

        eDot = 0.5*(n*eye(3) + skewSymmetric(e))*w;
        nDot = -0.5*e'*w;

        dstate = [wDot;eDot;nDot];
    end

% predictor
wHatPrev = [xHatPrev(1);xHatPrev(2);xHatPrev(3)];
eHatPrev = [xHatPrev(4);xHatPrev(5);xHatPrev(6)];
nHatPrev = xHatPrev(7);

rowA = [J\(-skewSymmetric(wHatPrev)*J + skewSymmetric(J*wHatPrev)), zeros(3,3), zeros(3,1)];
rowB = [0.5*(nHatPrev*eye(3) + skewSymmetric(eHatPrev)), -0.5*skewSymmetric(wHatPrev), 0.5*wHatPrev];
rowC = [-0.5*eHatPrev', -0.5*wHatPrev', 0];

FPrev = eye(7) + delT * [rowA;rowB;rowC];
LPrev = [delT*inv(J);zeros(3,3);zeros(1,3)];

f_hat_prev = TFM([wHatPrev;eHatPrev;nHatPrev], 0, J);

xHatMinus = xHatPrev + f_hat_prev*delT;
wHatMinus = xHatMinus(1:3);
eHatMinus = xHatMinus(4:6);
nHatMinus = xHatMinus(7);

PkMinus = FPrev*PPrev*FPrev' + LPrev*Q*LPrev';

PkMinus_1 = PkMinus(:,1:3);
PkMinus_2 = PkMinus(:,4:7);

PkMinus_ww = PkMinus(1:3,1:3);
PkMinus_qw = PkMinus(4:7,1:3);
PkMinus_wq = PkMinus(1:3,4:7);
PkMinus_qq = PkMinus(4:7,4:7);
```

```matlab
% corrector
M = eye(6);
H = [zeros(3,3), Y_aa_q(bA, eHatMinus, nHatMinus);
       zeros(3,3), Y_aa_q(sA, eHatMinus, nHatMinus)];

W = H*PkMinus*H' + M*R*M';
K_w = PkMinus_1'*H'*inv(W);
h_y = H*xHatMinus;
wHat = wHatMinus + K_w*(y - h_y);

K_tilde = PkMinus_2'*H'*inv(W);
r_k_vec = y - h_y;
r_k_scl = r_k_vec'*inv(W)*r_k_vec;

q_tilde = [eHatMinus;nHatMinus] + K_tilde*r_k_vec;

K_qk = K_tilde + 1/r_k_scl * (1/norm(q_tilde) - 1)*q_tilde*r_k_vec'*inv(W);

qHat = [eHatMinus;nHatMinus] + K_qk*r_k_vec;

xHat = [wHat;qHat];

Pk_ww = PkMinus_ww - K_w*H*PkMinus_1 - PkMinus_1'*H'*K_w' + K_w*W*K_w';
Pk_wq = PkMinus_wq - K_w*H*PkMinus_2 - PkMinus_1'*H'*K_qk' + K_w*W*K_qk';
Pk_qw = PkMinus_qw - K_qk*H*PkMinus_1 - PkMinus_2'*H'*K_w' + K_qk*W*K_w';
Pk_qq = PkMinus_qq - K_qk*H*PkMinus_2 - PkMinus_2'*H'*K_qk' + K_qk*W*K_qk';

Pk = [Pk_ww, Pk_wq; Pk_qw, Pk_qq];

end
```

```
function [wDot, eDot, nDot] = fcn(T, J, w, e, n)

    function wx = skewSymmetric(w)
        wx = [0, -1*w(3), w(2);
             w(3), 0, -1*w(1);
             -1*w(2), w(1), 0];
    end

wDot = J\(T-skewSymmetric(w)*J*w);

eDot = 0.5*(n*eye(3) + skewSymmetric(e))*w;
nDot = -0.5*e'*w;

end
```