Will Wu ESE 4481

Homework 7

April 15, 2022

After successfully designing a Kalman Filter for state estimation in our last assignment, we now shift our attention to another vital component before our drone becomes airborne: the controller. In this assignment, we will design LQR controller/regulators that holds our quadcopter in equilibrium position. To design LQR controllers/regulators, we must first linearize our drone around a equilibrium condition in order to obtain a linear state space model. We then examined their controllability and observability to ensure that we can control these systems. To facilitate our design process, we split our equations up to form five separate linear state-space systems and designed separate regulators for each system.

The code and simulation used for this assignment, which can be found in the appendix of this document, is a collaborative effort between Will Wu, the author of this paper, and Jackson Cox. This report is written by the author alone.

1 Linearization and Equilibrium Conditions

1.1 Drone Dynamics

From homework 3, we obtain the following 12 state variables, state inputs and state-space equations in the form of $\dot{x} = f(x, u)$.

State variables:

- Position in earth frame, $P^E = \begin{bmatrix} P_N \\ P_E \\ P_D \end{bmatrix}$
- Velocity in body frame, $V^b = \begin{bmatrix} u \\ v \\ w \end{bmatrix}$
- Rotation from earth to body, $\Omega = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}$
- Body rotation rate, $\omega_{b/E} = \begin{bmatrix} p \\ q \\ r \end{bmatrix}$

We also set up a rotation matrix to help translate between earth and body frame. From earth to body frame, we have the following matrix:

$$R_E^b = R(\phi)R(\theta)R(\psi) \tag{1}$$

where

- $R(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix}$

- $R(\theta) = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$
- $R(\psi) = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Since the above matrices are all orthonormal, the rotation matrix from body to earth holds the following relationship:

$$R_b^E = (R_E^b)^{-1} = (R_E^b)^T = R^T(\psi)R^T(\theta)R^T(\phi) \tag{2}$$

We have four control variables, $\begin{bmatrix} T & A & E & R \end{bmatrix}$.

Based on conservation of force and momentum, we can come up with the following dynamics equations:

$$\dot{p}^E = V^E = R_E^b \cdot V^b \tag{3}$$

$$\dot{V}^b = \frac{1}{m}(F_T^b + F_A^b + F_G^b - \omega_{b/E} \times mV^b) \tag{4}$$

where

$$F_T^b = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \cdot T,\ F_A^b = -T \cdot K_c \cdot V^b,\ F_G^b = R_E^b \cdot \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} \tag{5}$$

$$\dot{\Theta} = H(\Theta)^{-1}\omega_{b/E} \tag{6}$$

where

$$H(\Theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi)\cos(\theta) \\ 0 & -\sin(\theta) & \cos(\phi)\cos(\theta) \end{bmatrix} \tag{7}$$

$$\dot{\omega}_{b/E} = J^{-1}(M_T^b + M_\Omega^b - \omega_{b/E} \times J\omega_{b/E}) \tag{8}$$

For simplicity, we ignore the gyroscopic moment term M_Ω^b for the remainder of this assignment.

1.2 Equilibrium Conditions

For our past assignments, we use the simplest equilibrium: hovering. However, because we are interested in a dynamic drone that can be steered, we will investigate a second type of equilibrium here: constant linear motion. According to Newton's First Law of Motion, any object are either stationary or moving in constant linear speed when there is no (net) force present. Combining this condition, we can say that when $V^E = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$, that is , when the drone is moving in the x direction with a constant speed of $1m/s$, the net force acting on the drone is equal to 0:

$$F_T^b + F_A^b + F_G^b = 0 \tag{9}$$

We assume that the drone is only moving in the x direction. Intuitively, the drone needs to maintain a pitch angle in the same axis in order to move forward, though the rotational rates of the drone in all directions should be zero. Lastly, since there is no rotation, all inputs except for thrust T should be zero. The drone needs a non-zero thrust to maintain airborne. Therefore, we assume that θ and T are nonzero and $V_x^e = 1$. After transforming all physical quantities into the body frame, solving for equation (9) yields the following:

$$\theta = -0.2119\, rads \tag{10}$$
$$T = gm\cos\theta \tag{11}$$

We will use the above equilibrium states and inputs to linearize our state space equations.

1.3 Linearization

Jacobian Linearization provides us with a way to approximate change in systems around a set of equilibrium points. It is calculated as follows:

$$\dot{\delta_x}(t) = \frac{\partial f}{\partial x}|_{x=x_{eq},t=t_{eq}}\delta_x(t) + \frac{\partial f}{\partial u}|_{x=x_{eq},u=u_{eq}}\delta_u(t) \tag{12}$$

where:

$$\delta_x(t) = x(t) - x_{eq} \tag{13}$$
$$\delta_u(t) = u(t) - u_{eq} \tag{14}$$

From calculating the system jacobians, we can deduce the A and B matrix required for a linear state space model.

$$A := \frac{\partial f}{\partial x}|_{x=x_{eq},t=t_{eq}}\delta_x(t) \in \mathbb{R}^{n\times n} \tag{15}$$
$$B := \frac{\partial f}{\partial u}|_{x=x_{eq},u=u_{eq}}\delta_u(t) \in \mathbb{R}^{n\times m} \tag{16}$$

where n denotes the number of states, and m denotes the number of inputs.

The calculation and specific matrices are covered by our MATLAB script, appended at the end of this report.

To facilitate our design process, we decouple the entire state space system into 5 separate and simpler linear systems, with each system containing a subset of state variables and the input that affects those state variables. After linearization, we separate our entire system into the following 5 systems:

- $\begin{bmatrix}\delta\dot{P_D}\\ \delta\dot{w}\end{bmatrix} = A_{\delta T}, \begin{bmatrix}\delta P_D\\ \delta w\end{bmatrix} + B_{\delta T}\delta T, \quad y = \begin{bmatrix}\delta P_D\\ \delta w\end{bmatrix}$
- $\begin{bmatrix}\delta\dot{v}\\ \delta\dot{\phi}\\ \delta\dot{p}\end{bmatrix} = A_{\delta A}\begin{bmatrix}\delta v\\ \delta\phi\\ \delta p\end{bmatrix} + B_{\delta A}\delta A, \quad y = \begin{bmatrix}\delta v\\ \delta\phi\\ \delta p\end{bmatrix}$
- $\begin{bmatrix}\delta\dot{u}\\ \delta\dot{\theta}\\ \delta\dot{q}\end{bmatrix} = A_{\delta E}\begin{bmatrix}\delta u\\ \delta\theta\\ \delta q\end{bmatrix} + B_{\delta E}\delta E, \quad y = \begin{bmatrix}\delta u\\ \delta\theta\\ \delta q\end{bmatrix}$
- $\begin{bmatrix}\delta\dot{P_N}\\ \delta\dot{P_E}\end{bmatrix} = A_{guid}\begin{bmatrix}\delta P_N\\ \delta P_E\end{bmatrix} + B_{guid}\begin{bmatrix}\delta u\\ \delta\psi\end{bmatrix}, \quad y = \begin{bmatrix}\delta P_N\\ \delta P_E\end{bmatrix}$

To construct the decoupled A matrices, we extract elements from our A and B matrix in the following fashion. Given a state space vector, $\begin{bmatrix}x_i\\ x_j\\ x_k\end{bmatrix}$, where x_i is the i^{th} element from our previous, 1x12 state vector. Therefore,

$$A = \begin{bmatrix}A_{ii} & A_{ij} & A_{ik}\\ A_{ji} & A_{jj} & A_{jk}\\ A_{ki} & A_{kj} & A_{kk}\end{bmatrix}, \; B = \begin{bmatrix}B_i^u\\ B_j^u\\ B_k^u\end{bmatrix} \tag{17}$$

where u is the control related to the state variable.

Since ψ and u are both states rather than control variables, the B_{guid} matrix will be extracted from matrix A. Since all system outputs are equal to its own state vector, the C matrix for each system will be an identity matrix of corresponding dimensions.

1.4 Controllability and Observability

To study the controllability and observability of each system, we can determine the row ranks of their corresponding Kalman controllability and observability matrix in the following form:

$$M_c = \begin{bmatrix}B & AB & A^2B & ... & A^{n-1}B\end{bmatrix} \tag{18}$$

$$M_o = \begin{bmatrix} C & CA & CA^2 & ... & CA^{n-1} \end{bmatrix}^T \tag{19}$$

After computing the corresponding matrices for each system, we conclude that all 5 systems are both controllable and observable. However, we cannot directly control the last system, where $x = \begin{bmatrix} \delta P_N \\ \delta P_E \end{bmatrix}$, since their control inputs are state variables, which are controlled by our four inputs T, A, E and R. We will leave them unregulated for now.

The results of these calculations can be found in the MATLAB script appended below.

2 Controller Design and Simulation

2.1 LQR Controller/Regulator

We choose to use LQR regulators to control our drone given their optimal nature. A Linear Quadratic Regulator (LQR) is a type of optimal feedback controller, whose state-feedback gain is calculated to minimizing the system's quadratic cost function. Given a state space system:

$$\dot{x} = Ax + Bu \tag{20}$$

The quadratic cost function is defined as:

$$J = \int_0^\infty x^T(\tau)Qx(\tau) + u^T(\tau)Ru(\tau)d\tau \tag{21}$$

By minimizing J, we can obtain the gain K as the solution to this linearization problem. Therefore, this problem is now formulated as:

$$J(x_0) = \min_{u(t), t\geq 0} \int_0^\infty x^T(\tau)Qx(\tau) + u^T(\tau)Ru(\tau)d\tau \tag{22}$$

The solution to this optimization problem involves solving for the algebraic Riccati equation, which is beyond the scope of this report. We will use MATLAB to calculate the solution in the following sections.

We decided to prioritize position and rotation tracking for the all four systems. Therefore, we used the following Q penalty matrices.

- For system $\begin{bmatrix} P_D \\ w \end{bmatrix}$, $Q = \begin{bmatrix} 50 & 0 \\ 0 & 1 \end{bmatrix}$
- For system $\begin{bmatrix} v \\ \theta \\ q \end{bmatrix}$, $Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 50 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

- For system $\begin{bmatrix} u \\ \phi \\ p \end{bmatrix}$, $Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 50 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
- For system $\begin{bmatrix} \psi \\ r \end{bmatrix}$, $Q = \begin{bmatrix} 50 & 0 \\ 0 & 1 \end{bmatrix}$

Since all our subsystems are SIMO (single-input, multi-output) systems, $R \in \mathbb{R}$. We set $R = 1$ for all regulators for now.

2.2 Simulation

We modelled the system and controllers using Simulink. Below is an example system model for state variable pair u, $theta$ and q. The complete controller can be found in the appendix.

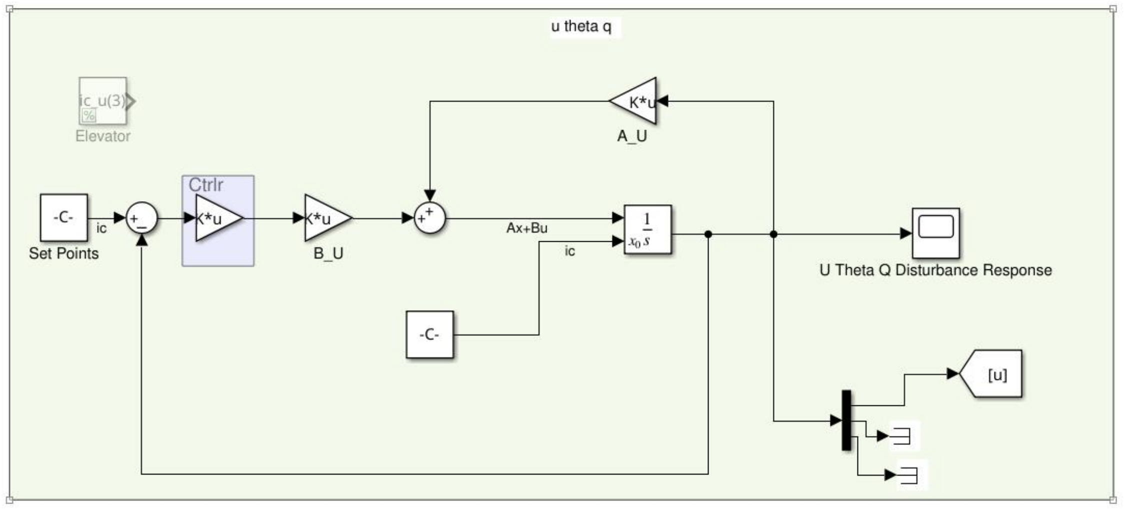
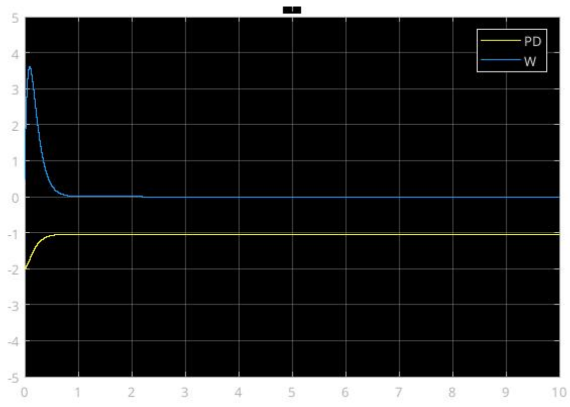


Figure 1: Sample System Block Diagram

To test the performance of our regulators, we initiated the drone in non-equilibrium initial conditions. In the future, we will further the controller robustness by adding in process noise.

2.3 Result Analysis

The simulation result for each subsystem can be found in the figure below:



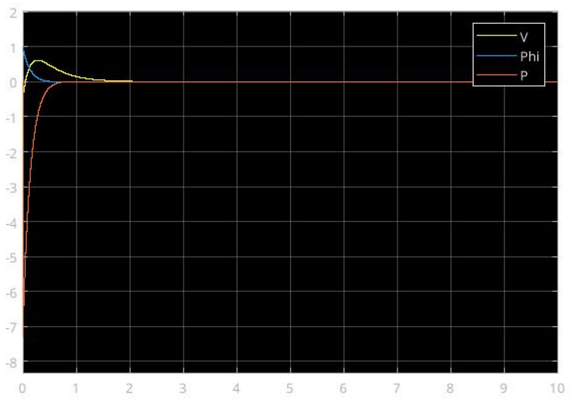
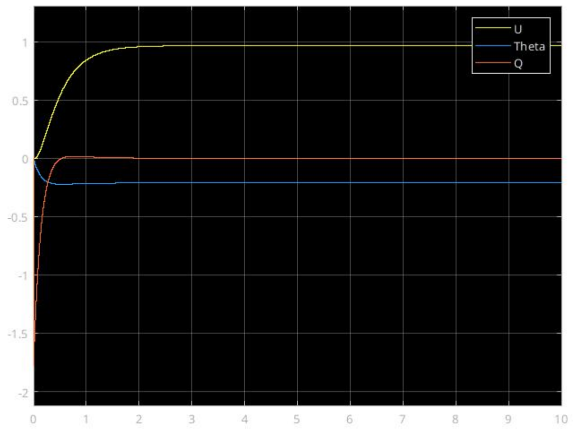


Figure 2: Simulation Results for P_D, w(left) and v, ϕ and p (right)



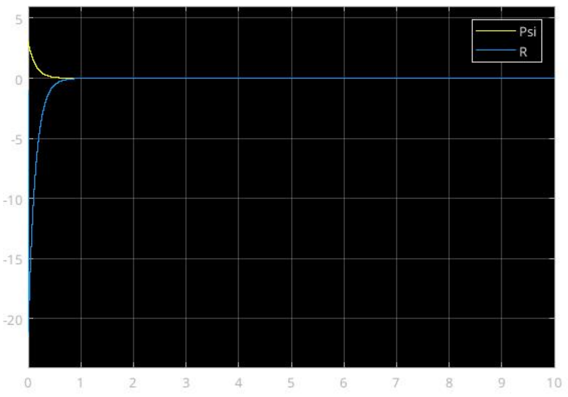


Figure 3: Simulation Results for u, θ and q(left) and ψ and r(right)

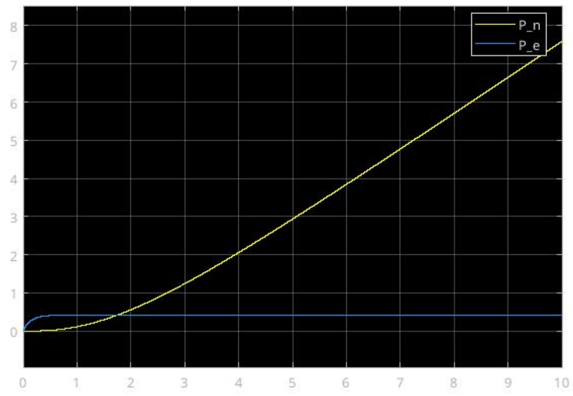


Figure 4: Simulation Results for P_n, and P_e (not directly regulated)

Overall, we can see that our LQR regulators are able to regulate the drone and asymptotically keep the drone stabilized under equilibrium conditions. Since the drone is moving in constant-linear motion, we can note, in figure 4, that the drone's position, after overcoming the initial disturbance, the position of the drone in the N direction linearly increases.

It is important to consider the requested control input and whether the actual aircraft can provide sufficient input. Therefore, in the next assignment we will analyze the gain and phase margin of the aircraft, as well as the model control saturation and investigate potential hazards such as integrator wind-up.

References

[1] Aircraft Control and Simulation, Stevens, Lewis & Johnson

[2] Homework 2, 3 Report

[3] Lecture Slides

Appendix: Simulation Materials

2.4 Matlab Script

Listing 1: Statistics Analysis Script

```matlab
% Jackson Cox, Will Wu
% Assignment 7

clc, clear; close all;

%symbolic variables
syms phi theta psi;
%rotation matrices
roll_BE = simplify(inv([1 0 0; 0 cos(phi) sin(phi); 0 -sin(phi) cos(phi)]));
pitch_BE = simplify(inv([cos(theta) 0 -sin(theta); 0 1 0; sin(theta) 0 cos(theta);]));
yaw_BE = simplify(inv([cos(psi) sin(psi) 0; -sin(psi) cos(psi) 0; 0 0 1;]));
%initial condition solver
syms m g
grav_body = roll_BE'*pitch_BE'*yaw_BE'*[0;0;m*g];
grav_body = subs(grav_body, phi, 0);
vb_i = roll_BE'*pitch_BE'*yaw_BE'*[1;0;0];
vb_i = subs(vb_i,[phi psi],[0 0]);
syms u v w
vb = [u;v;w;];
pE_dot = yaw_BE*pitch_BE*roll_BE*vb;
syms p q r
w_BE =[p;q;r];
BigT = [phi;theta; psi];
H = [1 0 -sin(theta); 0 cos(phi) sin(phi)*cos(theta); 0 -sin(phi) cos(phi)*cos(theta)];
BigT_dot = simplify(inv(H)*w_BE);
syms v1 v2 v3 v4 rho ct D m g T A E R
FbT = [0;0;T];
FbA = -T*[0.22 0 0; 0 0.22 0; 0 0 0;]*vb;
FgB = roll_BE'*pitch_BE'*yaw_BE'*[0;0; m*g];
vb_dot = (1/m)*(FbT + FbA + FgB - cross(w_BE,m*vb)) ;
syms Jp n1 n2 n3 n4 Jxx Jyy Jzz
MbT = ([A;E;R]);
MbOmega = cross(w_BE, [0;0;Jp*2*pi*(n1-n2+n3-n4);]);
J = [Jxx 0 0; 0 Jyy 0; 0 0 Jzz];
w_BE_dot = inv(J)*(MbT +MbOmega -cross(w_BE,J*w_BE));

%linearization
f= [pE_dot;vb_dot(3);vb_dot(2);BigT_dot(1);w_BE_dot(1);vb_dot(1);BigT_dot(2);w_BE_dot(2);BigT_dot(3);w_BE_dot(3)
    ];
%state vector
syms pN pE pD
x_vec = [pN;pE;pD;w;v;phi;p;u;theta;q;psi;r;];
%input vector
u_vec = [T; A; E; R;];
```

```
%initial conditions
x0 = [0; 0; -1; sin(-.2119); 0; 0; 0; cos(.2119); -.2119; 0; 0; 0;]; %hover or initial velocity??? Ask about this
    .
xhover = [0; 0; -1; 0; 0; 0; 0; 0; 0; 0; 0; 0;];
ic_hover = [m*g;0;0;0]; %not sure how to code this up as 1 m/s of x-velocity
ic_u = [0.6522;0;0;0];
f_x = jacobian(f, x_vec);
f_u = jacobian(f,u_vec);
A = subs(f_x,x_vec, x0);
A = subs(A, u_vec, ic_u);
B = subs(f_u, x_vec, x0);
B = subs(B, u_vec, ic_u);
% add step where we plug in constants
A = subs(A, [g D ct rho m Jp],[9.81 2*.066 .08 1.225 .068 1.089*10^(-6)]);
B = subs(B, [m Jxx Jyy Jzz], [.068 .69e-4 .775e-4 1.5e-4]);


% Decoupling into 5 Different Systems
% Throttle, PD, and w
A_dT_PD = double(A(3:4,3:4));
B_dT_PD = double(B(3:4,1));
A_dt_v = double(A(5:7,5:7));
A_dt_u = double(A(8:10,8:10));
A_dt_psi = double(A(11:12,11:12));
B_dt_v = double(B(5:7, 2));
B_dt_u = double(B(8:10, 3));
B_dt_psi = double(B(11:12,4));
A_dt_PN = double(A(1:2,1:2));
B_dt_PN = double([A(1,8) A(1,11);A(2,8) A(2,11)]);

%% Controllability and Observability
syms s
% PD w
ctrb_PD = rref(ctrb(A_dT_PD, B_dT_PD));
obsv_PD = rref(obsv(A_dT_PD, eye(2)));
tf_PD = simplify(eye(2) * inv(s*eye(2) - A_dT_PD) * B_dT_PD);
% V Phi P
ctrb_v = rref(ctrb(A_dt_v, B_dt_v));
obsv_v = rref(ctrb(A_dt_v, eye(3)));
tf_v = simplify(eye(3) * inv(s*eye(3) - A_dt_v) * B_dt_v);
% u theta q
ctrb_u = rref(ctrb(A_dt_u, B_dt_u));
obsv_u = rref(obsv(A_dt_u, eye(3)));
% Psi R
ctrb_psi = rref(ctrb(A_dt_psi, B_dt_psi));
obsv_psi = rref(ctrb(A_dt_psi, eye(2)));
tf_psi = simplify(eye(2) * inv(s*eye(2) - A_dt_psi) * B_dt_psi);
% PN PE
ctrb_pn = rref(ctrb(A_dt_PN, B_dt_PN));
obsv_pn = rref(ctrb(A_dt_PN, eye(2)));
tf_pn = simplify(eye(2) * inv(s*eye(2) - A_dt_PN) * B_dt_PN);

%% Calculating LQR
[K_PD, S] = lqr(A_dT_PD, B_dT_PD, [50 0; 0 1], 1);
[K_v, S] = lqr(A_dt_v, B_dt_v, [1 0 0; 0 50 0; 0 0 1], 1);
[K_u, S] = lqr(A_dt_u, B_dt_u, [1 0 0; 0 1 0; 0 0 50], 1);
[K_psi, S] = lqr(A_dt_psi, B_dt_psi, [50 0; 0 1], 1);
```

2.5 Simulink Block Diagram

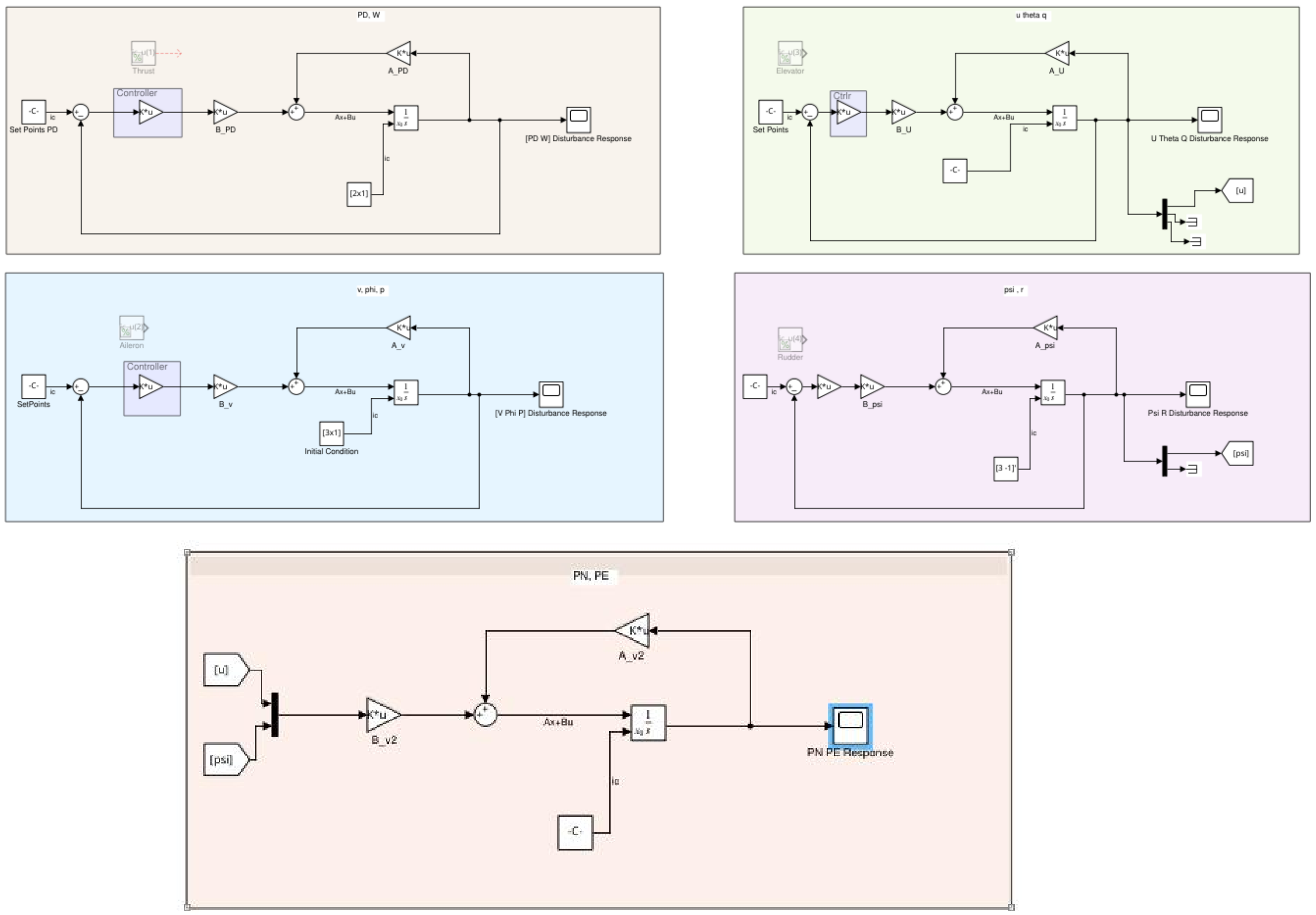


Figure 5: Subsystem and Controller Blocks