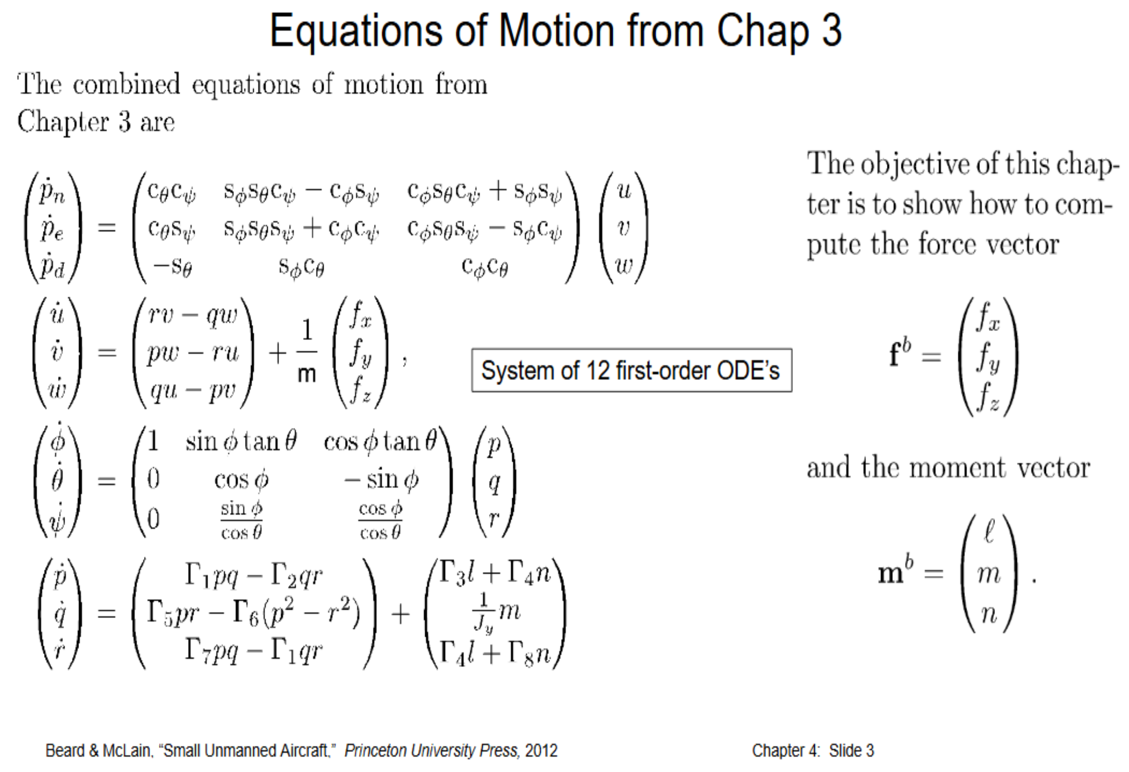
ESE 4481 - Final Report

David Young

NO-LINEAR MODEL

The nonlinear model that was used was found in forces and moments lecture:



Equations of Motion from Chap 3

The combined equations of motion from Chapter 3 are

$$\begin{pmatrix}\dot{p}_n\\ \dot{p}_e\\ \dot{p}_d\end{pmatrix} = \begin{pmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{pmatrix}\begin{pmatrix}u\\ v\\ w\end{pmatrix}$$

$$\begin{pmatrix}\dot{u}\\ \dot{v}\\ \dot{w}\end{pmatrix} = \begin{pmatrix}rv - qw\\ pw - ru\\ qu - pv\end{pmatrix} + \frac{1}{m}\begin{pmatrix}f_x\\ f_y\\ f_z\end{pmatrix},$$

System of 12 first-order ODE's

$$\begin{pmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{pmatrix} = \begin{pmatrix}1 & \sin\phi\tan\theta & \cos\phi\tan\theta\\ 0 & \cos\phi & -\sin\phi\\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta}\end{pmatrix}\begin{pmatrix}p\\ q\\ r\end{pmatrix}$$

$$\begin{pmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{pmatrix} = \begin{pmatrix}\Gamma_1 pq - \Gamma_2 qr\\ \Gamma_5 pr - \Gamma_6(p^2 - r^2)\\ \Gamma_7 pq - \Gamma_1 qr\end{pmatrix} + \begin{pmatrix}\Gamma_3 l + \Gamma_4 n\\ \frac{1}{J_y} m\\ \Gamma_4 l + \Gamma_8 n\end{pmatrix}$$

The objective of this chapter is to show how to compute the force vector

$$\mathbf{f}^b = \begin{pmatrix}f_x\\ f_y\\ f_z\end{pmatrix}$$

and the moment vector

$$\mathbf{m}^b = \begin{pmatrix}\ell\\ m\\ n\end{pmatrix}.$$

Beard & McLain, "Small Unmanned Aircraft," *Princeton University Press*, 2012 Chapter 4: Slide 3

Fig. 1. Nonlinear Model Used for the Drone

where the states and the inputs were as follows:

- **States:**
  - p_n: Position North (inertial frame)
  - p_e: Position East (inertial frame)
  - p_d: Position Down (inertial frame)
  - u: Body-frame velocity in x-direction (forward)
  - v: Body-frame velocity in y-direction (sideways)
  - w: Body-frame velocity in z-direction (vertical)
  - ϕ: Roll angle
  - θ: Pitch angle
  - ψ: Yaw angle
  - p: Roll rate
  - q: Pitch rate
  - r: Yaw rate
- **Inputs:**
  - f_x: Force in body x-direction
  - f_y: Force in body y-direction
  - f_z: Force in body z-direction
  - l: Roll moment (torque)
  - m: Pitch moment (torque)
  - n: Yaw moment (torque)

Trim States and Controls

The trim states and controls is the states and the controls that keep the system at an equilibrium point, mathematically, this means that all of the first derivatives are zero.

$$\dot{\vec{x}} = \vec{0}, \tag{1}$$

where $\dot{\vec{x}}$ is the vector that contains how the states change based off the current states.

In this project, there are two equilibrium points: when the drone is at rest and when the drone is hovering 0.5 meters off the ground. Because equilibrium where the drone is at rest is the trivial case where all of the states and controls are zero, I will ignore this equilibrium. Thus, the states of the system at equilibrium (trim states) are found when the positions, velocities, Euler angles (roll, pitch, and yaw), rate of change of the Euler angles are all zero. In this case, all of the states at the hover equilibrium will be zero except for the "pd" state which is equal to -0.5 meters. It is negative because the state pd points into the ground. For the trim controls, I do not want any force or moments except in the up and down directions. In this case, if I want the drone to hover, then the force in the up and down directions must equal the force of gravity; thus,

$$fz = -m \cdot g \tag{2}$$

where m is the mass of the drone (0.033 kg) and g is the acceleration due to gravity (9.8 $\frac{m}{s^2}$).

$$\underbrace{\begin{bmatrix} p_n \\ p_e \\ p_d \\ u \\ v \\ w \\ \phi \\ \theta \\ \psi \\ p \\ q \\ r \\ f_x \\ f_y \\ f_z \\ l \\ m \\ n \end{bmatrix}}_{\text{States and Inputs}} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ -0.5 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -mg \\ 0 \\ 0 \\ 0 \end{bmatrix}}_{\text{Trim Values}} \tag{3}$$

Linearization

To linearize the non-linear model, I will first turn the equations seen in figure 1 into one big matrix as seen below. Then, I will take the Jacobian of the matrix with respect to the states in the same order as seen in the bulleted list above, meaning I will take the Jacobian with respect to state pn, then pe, and so on. After obtaining the Jacobian with respect to the states, I evaluated Jacobian at the trim values, obtaining δA:

$$\delta A = \begin{bmatrix} 0&0&0&1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0 \end{bmatrix} \tag{4}$$

The same process was repeated to obtain the B matrix but instead of taking the Jacobian with respect to the states, I took the Jacobian with respect to the six inputs and evaluated the Jacobian at the trim conditions.

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 30.30 & 0 & 0 & 0 & 0 & 0 \\ 0 & 30.30 & 0 & 0 & 0 & 0 \\ 0 & 0 & 30.30 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.06042 & 0 & 1.486 \\ 0 & 0 & 0 & 0 & 0.06002 & 0 \\ 0 & 0 & 0 & 1.486 & 0 & 34.22 \end{bmatrix} \tag{5}$$

CONTROLLABILITY

The controllability matrix was calculated as follows:

$$\mathcal{C} = \begin{bmatrix} B & AB & A^2B & \cdots & A^{n-1}B \end{bmatrix} \tag{6}$$

If the rank of the controllability matrix has full row rank, meaning that the rank of the matrix is equal to the number of rows, then the system is controllable. Using Matlab's rank() function, I found that the rank of the matrix was 12, meaning that the system is controllable.

EXPLANATION OF CONTROL DESIGN TECHNIQUE AND METHODOLOGY OF GAIN SELECTION

The controller u, has four outputs: fz, L, M, and N. where fz is the force pointing in the down direction, L is the moment about the axis that goes straight the front and back of the drone, M is the moment about the axis that goes through the left and right of the drone, and N is the moment about the z axis of the drone which goes through the top and the bottom of the drone. All positive moments are determined by the right hand rule according to the NED orientation.

The four controller were obtained using the cascaded controller method. The following figures shows the block diagrams of the four controllers.

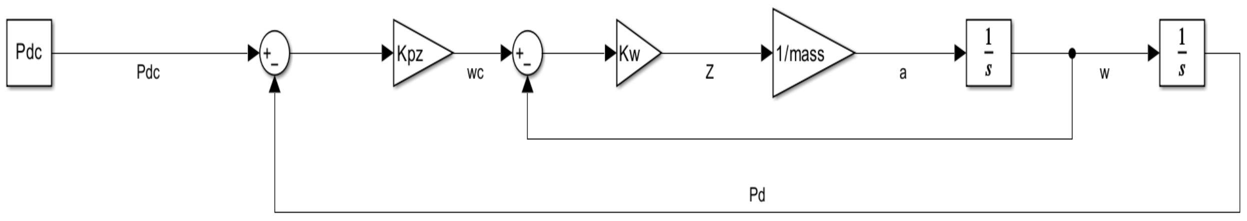


Fig. 2. Z Controller Block Diagram

where Kw is the gain that is multiplied with the error between the state w (the velocity in the body frame's z axis) and the commanded w (wc). Kpz is the gain that is multiplied by the error between the state Pd and the commanded Pd (Pdc).

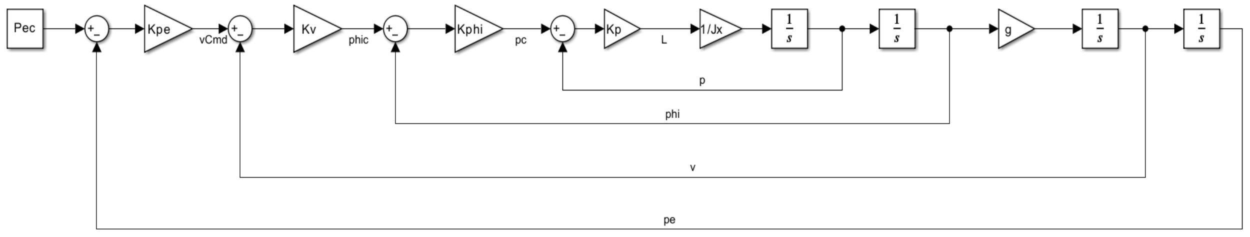


Fig. 3. L Controller Block Diagram

where Kp is the gain that is multiplied by the error between the state P (the roll rate) and the commanded roll rate (Pc). Kphi is the gain that is multiplied by the error between the state ϕ (the pitch angle) and the commanded pitch angle (phiCmd).

Kv is the gain that is multiplied by the error between the state v (the velocity in the east direction in the body frame) and the commanded v (vCmd). Kpe is the gain that is multipiled by the error between the state Pe (Position of the drone in the inertial frame). Lastly, Jx is the Jx element of the inertial tensor of the drone given by the following matrix.

$$\text{InertMat} = \begin{bmatrix} J_x & J_{xy} & J_{xz} \\ J_{xy} & J_y & J_{yz} \\ J_{xz} & J_{yz} & J_z \end{bmatrix} = \begin{bmatrix} 16.57 & 0.83 & 0.72 \\ 0.83 & 16.66 & 1.80 \\ 0.72 & 1.80 & 29.26 \end{bmatrix} \tag{7}$$

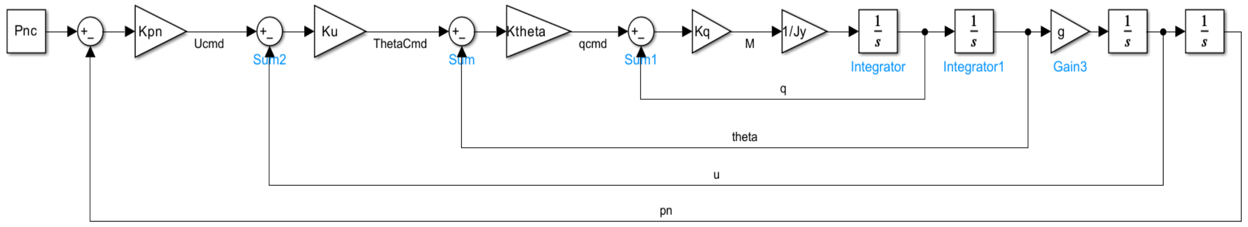


Fig. 4. M Controller Block Diagram

where Kq is the gain that is multiplied by the error between the state q (pitch rate in the body frame) and q commanded. Ktheta is the gain that is multiplied by the error between the state theta (the pitch angle in the body frame) and pitch angle commanded. Ku is the gain that is multiplied by the state u (the velocity of the drone in the forward direction in the body frame) and u commanded. Kpn is the gain that is multiplied by the error between the state Pn (the position of the drone in the north direction in the inertial frame) and Pn commanded. Lastly, Jy is element (2,2) in the inertial matrix.

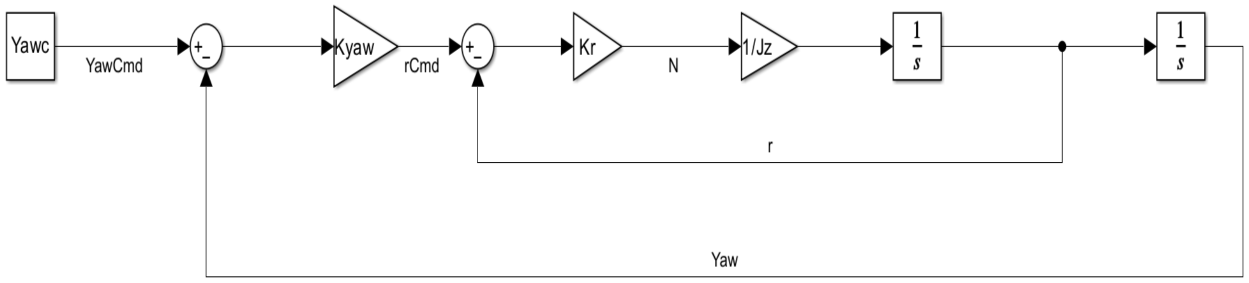


Fig. 5. N Controller Block Diagram

where Kr is the gain that is multiplied by the error between the state r (yaw rate in the body frame) and r commanded. Kyaw is the gain that is multiplied by the error between the state ψ (the yaw angle in the body frame) and ψ_{cmd}. Lastly, Jz is element (3,3) in the inertial matrix.

For each of these controllers, the gains were selected using the following steps.

For the Z controller, the drone provides measurements at a rate of 100 Hz for force measurements. The cascaded controller starts with the innermost loop, after which, we build outward. If we look at the closed loop transfer function of the innermost loop, I see that the transfer function is

$$TF_{inner} = \frac{Kw/m}{s + Kw/m} \tag{8}$$

By changing Kw, I change the location of the pole and thus, the corner frequency of the loop. To attenuate measurement noise, we want the corner frequency of the inner loop to be at most 10 times less than the measurement frequency 100 Hz. This means that I want the corner frequency to be at 10 Hz. I solved the following equation to find Kw:

$$\frac{kw}{m} = 10 * (2 * \pi) \tag{9}$$

There is an extra $2 \cdot \pi$ to convert from Hz to Rad/s.

Then for the outer loop, I treat the innermost loop to have a transfer function of 1, a wire, because I will choose a Kpz that will be 5 times slower that the innermost loop. The transfer function of the outer loop is then

$$TF_{second} = \frac{Kpz}{s + Kpz} \tag{10}$$

As mentioned above, I want the corner frequency of the outer loop to be 5 times less than the inner loop. I solved the rather simple equation to get the gain for Kpz.

$$Kpz = 5 * (2 * \pi) \tag{11}$$

The following list shows the gains for the Z controller.

- Mass of the drone: $m = 0.033$
- Gravity: $g = 9.8$
- Inner Loop Gain (Vertical velocity control):
  $K_w = 10 \cdot (2\pi) \cdot m = 2.0734$
- Second Loop Gain (Altitude/position control):
  $K_{pz} = 2 \cdot (2\pi) = 12.5664$

For the moment controllers (L, M, and N), the drone provides measurements at a rate of 500 Hz. The same design approach used for the Z controller was applied: the innermost loop operates at one-tenth the sampling rate, and each successive outer loop runs at one-fifth the bandwidth of the loop it encloses. THe following list gives the gains for each of the moment controllers.

L Controller gains:

- Inner Loop Gain (Angular rate control):
  $K_p = J_x \cdot 50 \cdot 2\pi = 16.57 \cdot 314.159 = 5203.94$
- Second Loop Gain (Roll angle control):
  $K_\phi = 10 \cdot 2\pi = 62.8319$
- Third Loop Gain (Lateral velocity control):
  $K_v = \frac{2 \cdot 2\pi}{g} = \frac{12.5664}{9.8} = 1.2812$
- Fourth Loop Gain (Lateral position control):
  $K_{pe} = \frac{2}{5} \cdot 2\pi = 2.5133$

M Controller gains:

- Inner Loop Gain (Angular rate control):
  $K_q = 50 \cdot 2\pi \cdot J_z = 314.159 \cdot 29.26 = 9195.13$
- Second Loop Gain (Pitch angle control):
  $K_\theta = 10 \cdot 2\pi = 62.8319$
- Third Loop Gain (Longitudinal velocity control):
  $K_u = \frac{2 \cdot 2\pi}{g} = \frac{12.5664}{9.8} = 1.2812$
- Fourth Loop Gain (Longitudinal position control):
  $K_{pn} = \frac{2}{5} \cdot 2\pi = 2.5133$

N Controller gains

- Inner Loop Gain (Yaw rate control):
  $K_r = 50 \cdot 2\pi \cdot J_z = 314.159 \cdot 29.26 = 9195.13$
- Second Loop Gain (Yaw angle control):
  $K_{\text{yaw}} = 5 \cdot 2\pi = 31.4159$

Diagram and Explanation of Control System Architecture

The following figure shows the block diagram that simulates the drone but without the actuator modeled, meaning that I am assuming that the actuator provides the exact forces that the controller requests (fz, L, M, and N).

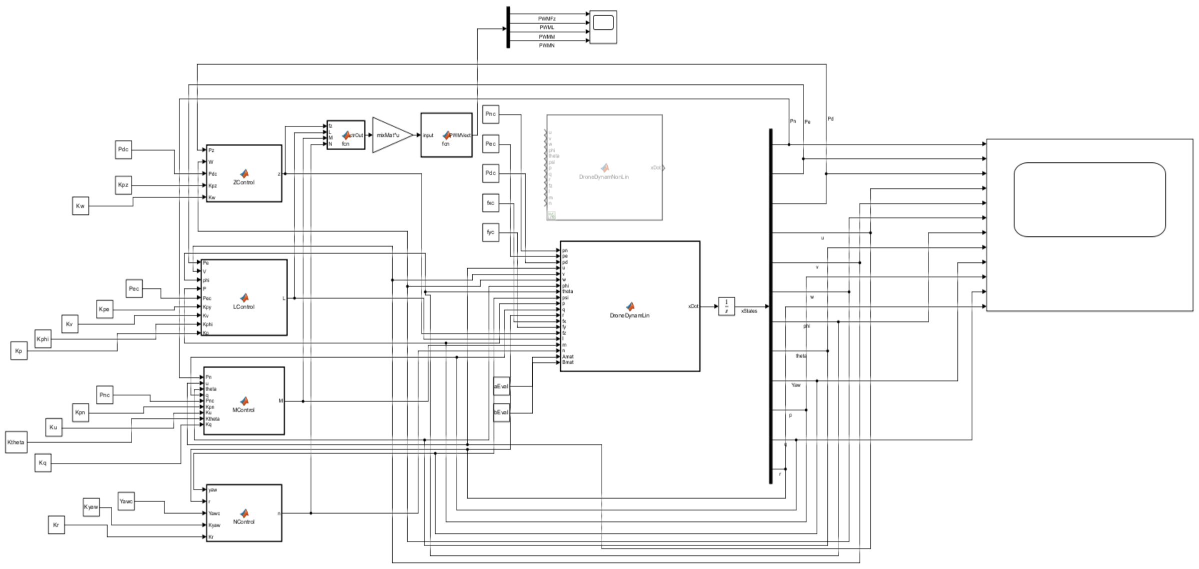


Fig. 6. Simulink Block Diagram of Linearized Drone Without Actuators.

The following figures shows a close-up of each of the blocks in the architecture.

The linearized drone dynamics block.

This block holds the linearized system dynamics.

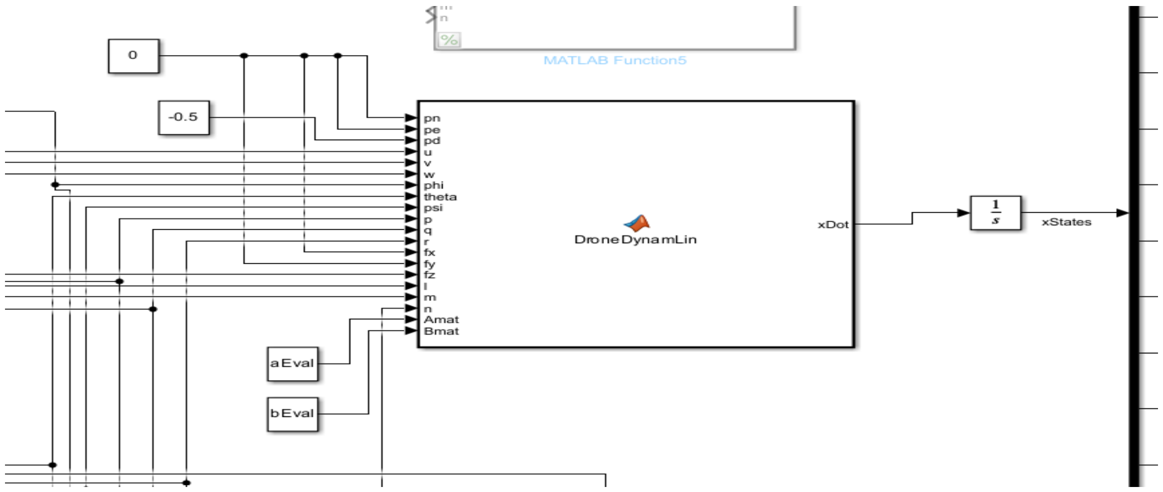
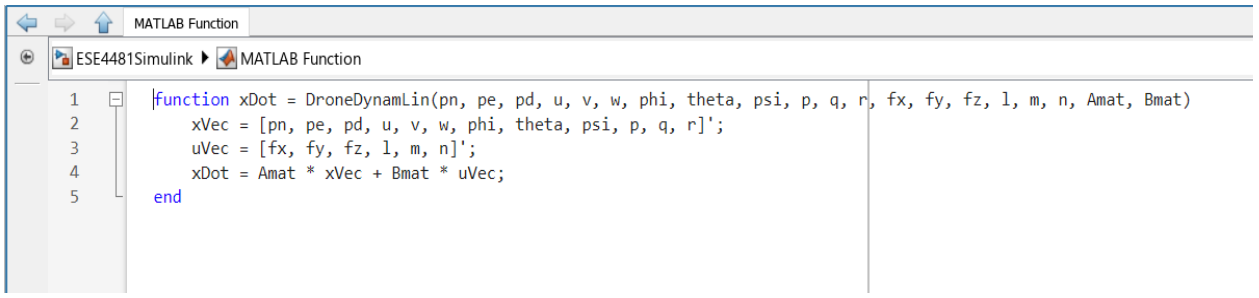


Fig. 7. Drone Dynamics Block



```
function xDot = DroneDynamLin(pn, pe, pd, u, v, w, phi, theta, psi, p, q, r, fx, fy, fz, l, m, n, Amat, Bmat)
    xVec = [pn, pe, pd, u, v, w, phi, theta, psi, p, q, r]';
    uVec = [fx, fy, fz, l, m, n]';
    xDot = Amat * xVec + Bmat * uVec;
end
```

Fig. 8. Function within Drone Dynamics Block

The inputs into the linearized drone dynamics block are the states, the controllers, and the A and B matrices. The output is a vector that contains the dynamics of how the states change given the current states.

Z Controller

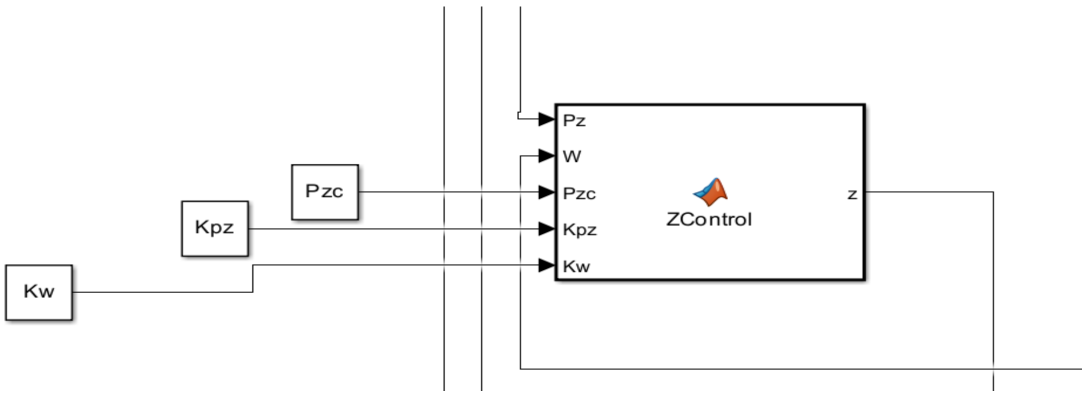
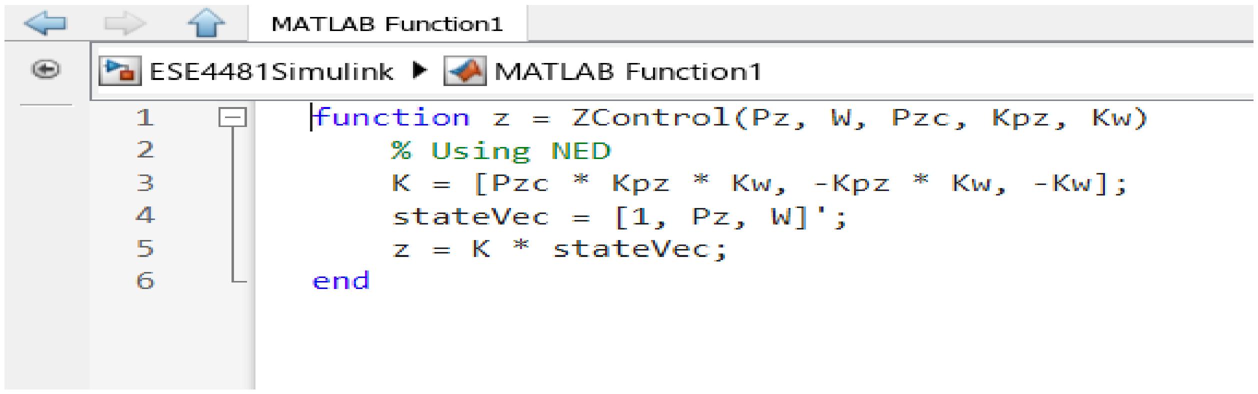


Fig. 9. Z Controller Block



```
function z = ZControl(Pz, W, Pzc, Kpz, Kw)
    % Using NED
    K = [Pzc * Kpz * Kw, -Kpz * Kw, -Kw];
    stateVec = [1, Pz, W]';
    z = K * stateVec;
end
```

Fig. 10. Z Controller Block Function

The inputs into the Zcontroller are the gains: Kpz, Kw; the command Pzc = -0.5 meters (because the Pz points downwards so a negative Pz value means up to the sky); and the states Pz and w. The output is a force in the z-axis. Z is obtained by algebraically solving for Z from figure 2.

L Controller

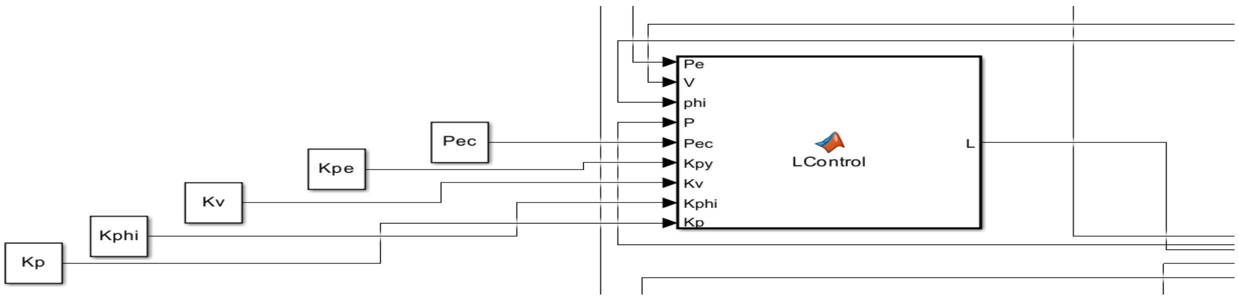
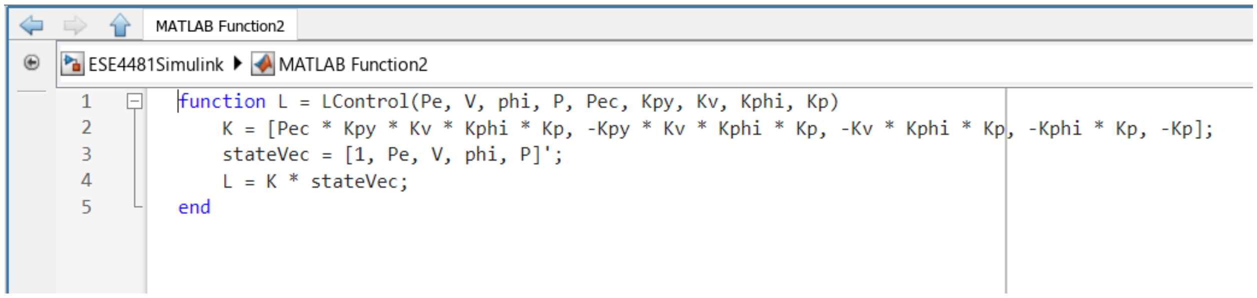


Fig. 11. L Controller Block



```
function L = LControl(Pe, V, phi, P, Pec, Kpy, Kv, Kphi, Kp)
    K = [Pec * Kpy * Kv * Kphi * Kp, -Kpy * Kv * Kphi * Kp, -Kv * Kphi * Kp, -Kphi * Kp, -Kp];
    stateVec = [1, Pe, V, phi, P]';
    L = K * stateVec;
end
```

Fig. 12. L Controller Block Function

The inputs are the gains:Kp, Kphi, Kv, Kpy; the command in the east direction (body frame): Pec = 0; and the states: Pe, v, ϕ, and p. The output is the moment about the longitudinal axis with the right hand rule pointing towards the front of the drone.

M Controller

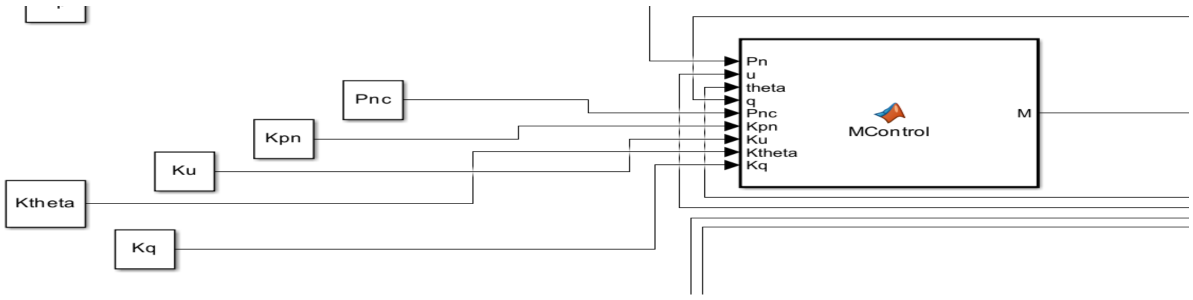
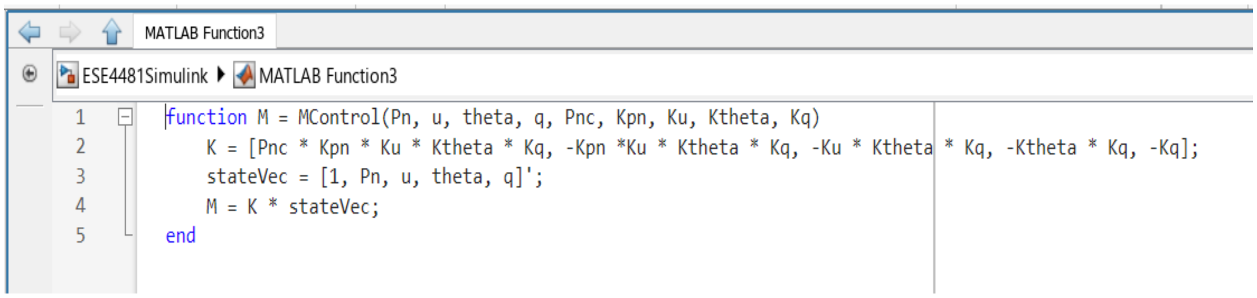


Fig. 13. M Controller Block



```
function M = MControl(Pn, u, theta, q, Pnc, Kpn, Ku, Ktheta, Kq)
    K = [Pnc * Kpn * Ku * Ktheta * Kq, -Kpn *Ku * Ktheta * Kq, -Ku * Ktheta * Kq, -Ktheta * Kq, -Kq];
    stateVec = [1, Pn, u, theta, q]';
    M = K * stateVec;
end
```

Fig. 14. M Controller Block Function

The inputs are the gains:Kq, Ktheta, Ku, and Kpn; the command in the north direction (body frame): Pnc = 0; and the states: Pn, u, theta, and q. The output is the moment about lateral axis with the right hand rule pointing towards east of the drone.

N Controller

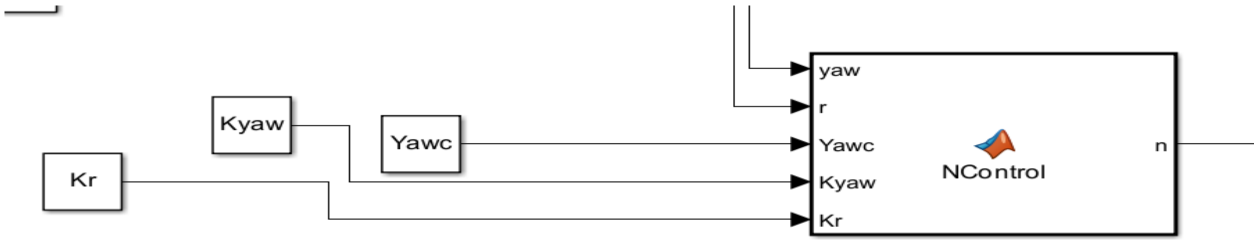
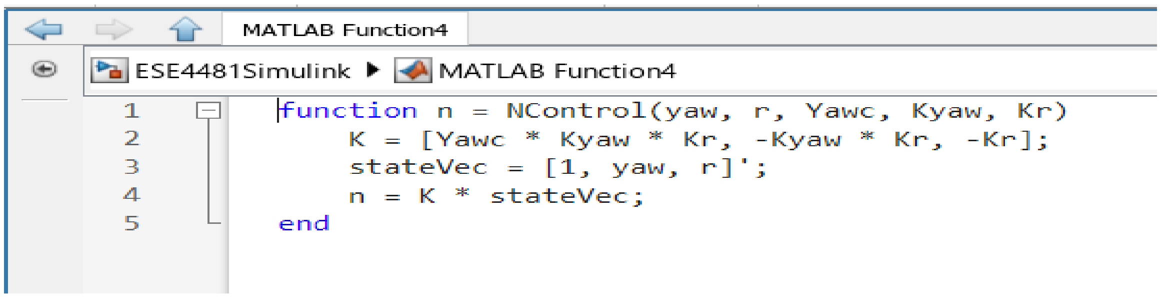


Fig. 15. N Controller Block



```
function n = NControl(yaw, r, Yawc, Kyaw, Kr)
    K = [Yawc * Kyaw * Kr, -Kyaw * Kr, -Kr];
    stateVec = [1, yaw, r]';
    n = K * stateVec;
end
```

Fig. 16. N Controller Block Function

The inputs are gains: Kr and Kyaw; the yaw angle command ψ_c = 0; and the states: ψ and r. The output is the moment about the drone's z axis with the right hand rule pointing towards the ground.

PWM From Controller Output

The following figure shows the blocks used to convert the forces requested by the controller (fz, L, M, and N) into PWM.

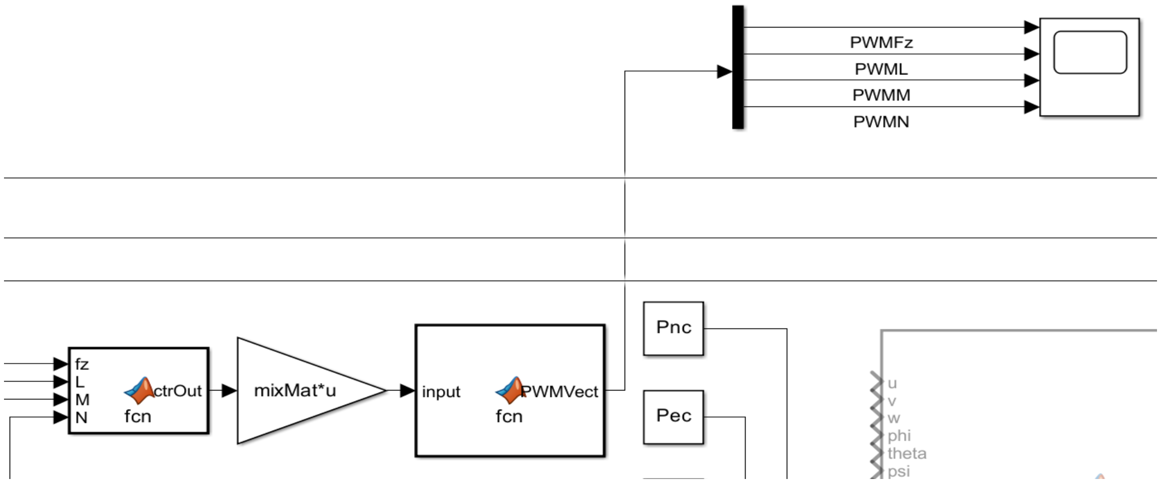


Fig. 17. Blocks that convert force to PWM

This section was used to partially model the actuator without actually implementing it into the system and to test the equations used to convert a force into a PWM. I will talk more about the actuator being implemented into the system in the Mixer section.

Linearized Drone System Step Response

For get the step response of the system, I set all of the input commands to zero Pdc = Pec = Pnc = Yawc = 0 and input a step input into one of these four commands. This gave me four step response plots as seen below.

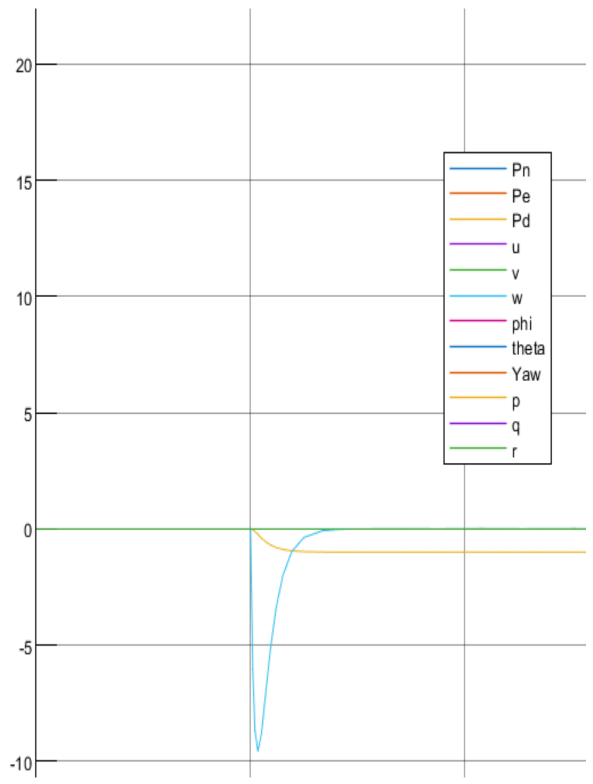


Fig. 18. System Response for Pdc = -Step

From the figure above, we can see that for a negative step input into Pdc, the state Pd goes to -0.5 meters and there is a velocity in the down direction as seen by the spike in the w state. All of these are to be expected.

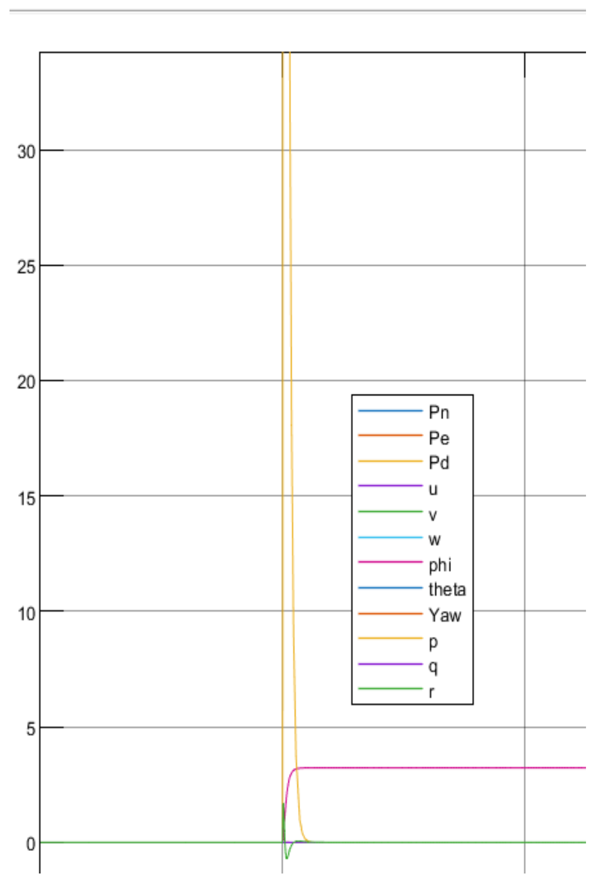


Fig. 19. Step Response for Pec = 1

From the figure above, state Pe goes to around 3 meters while the p state spikes which makes sense because the drone is rolling to move east. While the Pe state goes a little higher than 1 meter, the system response shows that it can stay at a constant position even though it may be slightly off.

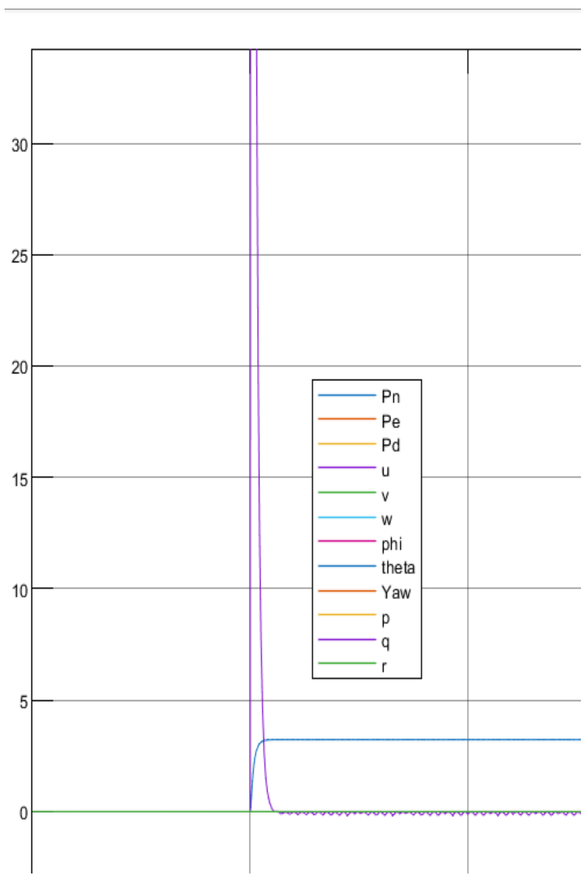


Fig. 20. Step Response for Pen = 1

From the figure above, state Pn goes to around 3 meters while the q state spikes which makes sense because the drone is pitch to move forward. While the Pn state goes a little higher than 1 meter, the system response shows that it can stay at a constant position even though it may be slightly off.

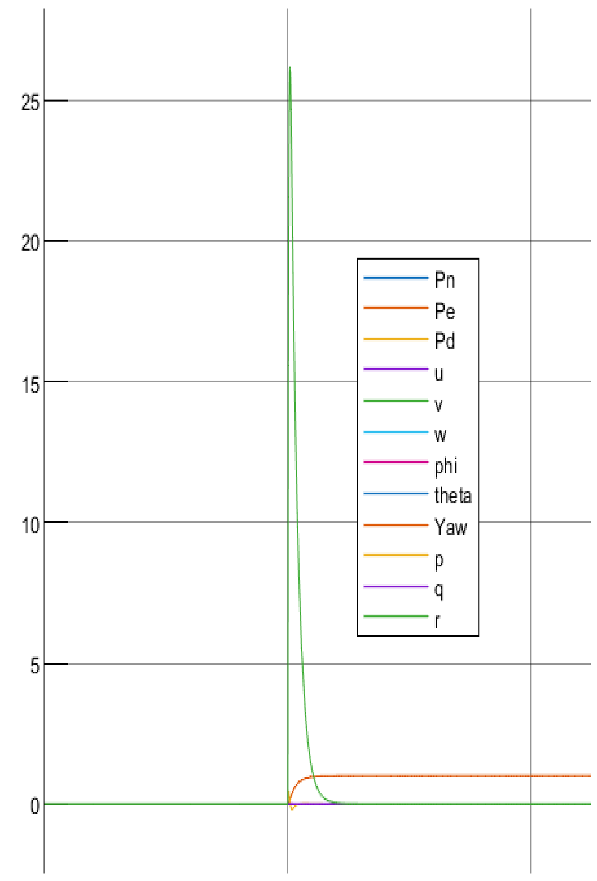


Fig. 21. Step Response for Yawc = 1

From figure above, the yaw state goes to around 1 while the r state spikes this makes sense since the yaw angle is changing.

Derivation of the Mixer

The equation that I used to obtain the mixer matrix M is as follows:

$$\begin{bmatrix} \text{PWM}_1 \\ \text{PWM}_2 \\ \text{PWM}_3 \\ \text{PWM}_4 \end{bmatrix} = M \begin{bmatrix} Z \\ L \\ M \\ N \end{bmatrix} \tag{12}$$

Z, L, M, and N are the forces given from the controller. If there is purely thrust in the z direction, then I want each motor to bear a fourth of the total thrust. Therefore, for the first column of M, I will have only -0.25. Once again, its -0.25 rather than 0.25 because my z direction is pointing down due to using NED body frame.

$$M_{\text{Col1}} = \begin{bmatrix} -0.25 \\ -0.25 \\ -0.25 \\ -0.25 \end{bmatrix} \tag{13}$$

For L and M, I want each motor to bear a fourth of the load, but for a positive L, motors 3 and 4 must spin faster and thus a larger positive number than motors 1 and 2.

$$M_{\text{Col2}} = \begin{bmatrix} 0.25 \\ 0.25 \\ 0.5 \\ 0.5 \end{bmatrix} \tag{14}$$

For a positive M, motors 1 and 4 must spin faster so they have a larger negative number.

$$M_{\text{Col3}} = \begin{bmatrix} 0.5 \\ 0.25 \\ 0.25 \\ 0.5 \end{bmatrix} \tag{15}$$

For a positive N, motors 1 and 3 must spin faster than motors 2 and 4. This results in the drone turning clock wise.

$$M_{\text{Col4}} = \begin{bmatrix} 0.35 \\ 0.2 \\ 0.35 \\ 0.2 \end{bmatrix} \tag{16}$$

Together this gives:

$$M = \begin{bmatrix} -0.25 & 0.25 & 0.5 & 0.35 \\ -0.25 & 0.25 & 0.25 & 0.2 \\ -0.25 & 0.5 & 0.25 & 0.35 \\ -0.25 & 0.5 & 0.5 & 0.2 \end{bmatrix} \tag{17}$$

Success and Improvements

While some of the states were a little off during the step response, I'd say that I successfully implemented four cascaded controllers. While I was more comfortable with LQR controllers throughout the course, cascaded controllers were a mystery to me. However, with the help of the TA, I was able to understand and apply cascaded controllers.

An area of improvement is obtaining the Bode plots and properly implementing the model of the actuators into the system. I was able to get Bode plots from the cascaded controllers but I just had difficulty incorporating the actuator into the loops because without the actuators, all of my controllers had infinite gain margin and 90 degrees phase margins.