ME 3295 Quadcopter Project

Michael Bernard

2020-04-26

Introduction

In this report, we detail the development of a controller for the quadcopter shown in Figure 1. The four system inputs are height, and three Euler angles representing orientation. The inputs are step inputs. The controller design goals were to:

- stabilize the closed-loop sytem for each input,
- ensure zero steady-state error for each input,
- minimize settling time for each input

with a time-delay of 0.2 seconds.

We begin in the 6 Degree of Freedom Model section by detailing the 6 Degree of Freedom (6-DOF) model for an aircraft. The 6-DOF model's inputs are the three Cartesian force and moment components, but the vehicle can only control each of these indirectly via propeller rotation. In the Physical Forces and Moments section, we detail the relation between the propeller angular velocities and these force and moment components.

The controller inputs are the current height and Euler angle errors, and the outputs are the appropriate force and moment components to stabilize the system. In the Control Allocation section, we inverse the relations derived in the Physical Forces and Moments section to yield propeller angular velocities given the force and moment components.

In the Controller Design section, we describe the development of the controller using Simulink. We include output plots demonstrating our controller meets design requirements.

In each section, we include a Simulink implementation of that section's model. A fully connected model is included at the end of this report. The definition and numerical values for all variables used in each equation are given in Table 1 at the end of this report.

6 Degree of Freedom Model

Due to symmetry about the x and y axes of the quadcopter, the vehicle's inertia matrix is defined as

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} , \tag{1}$$

where $I_{xx} = I_{yy}$ due to symmetry.

The Cartesian components of force on the vehicle in the inertial frame are defined as follows:

$$\begin{aligned} \dot{U} &= RV - QW - g_D \sin(\theta) + \frac{(X_A + X_T)}{m} \\ \dot{V} &= -RU + PW + g_D \sin(\phi)\cos(\theta) + \frac{(Y_A + Y_T)}{m} \\ \dot{W} &= QU - PV + g_D \cos(\phi)\cos(\theta) + \frac{(Z_A + Z_T)}{m} \quad . \end{aligned}$$

Rather than define all variables following each equation, all variables are defined in Table 1 at the end of this report. Since the flight speed of the quadcopter is low, the aerodynamic drag forces X_A, Y_A, and Z_A can be considered negligible. If $X_A = Y_A = Z_A = 0$, the equations simplify to

$$\dot{U} = RV - QW - g_D \sin(\theta) + \frac{X_T}{m} \tag{2}$$

$$\dot{V} = -RU + PW + g_D \sin(\phi)\cos(\theta) + \frac{Y_T}{m} \tag{3}$$

$$\dot{W} = QU - PV + g_D \cos(\phi)\cos(\theta) + \frac{Z_T}{m} \quad . \tag{4}$$

Figures 2, 3, and 4 show the simplified equations (2), (3), and (4) above as implemented in Simulink. Figure 5 shows how these blocks are connected at a high-level in a Force Equations block in Simulink.

The Kinematic Equations for the Euler Angles of the vehicle in the inertial frame are as follows:

$$\dot{\phi} = P + \tan(\theta)(Q\sin(\phi) + R\cos(\phi)) \tag{5}$$

$$\dot{\theta} = Q\cos(\phi) - R\sin(\theta) \tag{6}$$

$$\dot{\psi} = \frac{Q\sin(\phi) + R\cos(\phi)}{\cos(\theta)} \quad . \tag{7}$$

Figures 6, 7, and 8 show an implementation of these equations in Simulink. Figure 9 shows how these blocks are connected at a high-level in a Kinematic Equations block.

The general 6-DOF Moment Equations in the inertial frame are as follows:

$$\begin{aligned}
\dot{P} &= \frac{I_{xz}[J_{xx} - I_{yy} + I_{zz}]PQ - [I_{zz}(I_{zz} - I_{yy}) + I_{xy}^2]QR + I_{zz}l + I_{xz}n}{\Gamma} \\
\dot{Q} &= \frac{(I_{zz} - I_{xx})PR - I_{xz}(P^2 - R^2) + m}{I_{yy}} \\
\dot{R} &= \frac{[(I_{xx} - I_{yy})I_{xx} + I_{xz}^2]PQ - I_{xz}[I_{xx} - I_{yy} + I_{zz}]QR + I_{xz}l + I_{xx}n}{\Gamma} \quad .
\end{aligned}$$

Examining the simplified inertia matrix in equation (1), the Moment Equations simplify to:

$$\dot{P} = \frac{-I_{zz}(I_{zz} - I_{yy})QR + I_{zz}l}{\Gamma} \tag{8}$$

$$\dot{Q} = \frac{(I_{zz} - I_{xx})PR + m}{I_{yy}} \tag{9}$$

$$\dot{R} = \frac{[(I_{xx} - I_{yy})I_{xx}]PQ + I_{xx}n}{\Gamma} \quad . \tag{10}$$

Note here that l, m, and n are the x, y, and z components of moment due to the propellers (l and m are not the length and mass values given in Table 1).

Figures 10, 11, and 12 show an implementation of the equations above in Simulink. Figure 13 shows how these blocks are connected at a high-level in a Moment Equations block.

We are concerned only with altitude control of the vehicle, not lateral control. As such, we use only the following Navigation Equation:

$$\dot{h} = U\sin(\theta) - V\sin(\phi)\cos(\theta) - W\cos(\phi)\cos(\theta) \quad . \tag{11}$$

Figure 14 shows an implementation of this equation in Simulink. The equation was multiplied by -1 in the implementation in order to reflect a positive height being upwards, which is opposite the convention in the equation. Figure 15 shows how the Force Equations, Kinematic Equations, Moment Equations, and Navigation Equation blocks are connected at a high-level in a 6-DOF Model block in Simulink.

Physical Forces and Moments

Each of the rotors induces a lifting force on the vehicle:

$$f_i = k\omega_i^2 \tag{12}$$

and a torque about the rotor axis: $\tau_{M_i} = b\omega_i^2 + I_M\dot{\omega}_i$, where ω is the rotor angular velocity. The value of $\dot{\omega}_i$ is considered small, and so the torque equation is simplified to

$$\tau_{M_i} = b\omega_i^2 \quad . \tag{13}$$

The 6-DOF model has six inputs: the x, y, and z thruster force components (X_T, Y_T, Z_T), and the x, y, and z moment components (l, m, n). The physical quadcopter generates these forces and moments using its four propellers.

These force and moment components are resolved in the quadcopter body frame. Since the propellers can only induce a force in the "up" direction (z) in the body frame, the x and y force components are equal to zero. The z force component is equal to the propeller lift constant k multiplied by the sum of each propeller's angular velocity squared:

$$X_T = Y_T = 0 \tag{14}$$

$$Z_T = k \sum_{i=1}^{4} \omega_i^2 \quad . \tag{15}$$

Examining Figure 1, the moment about the x axis is due to the net force of propellers 2 and 4, and the moment about the y axis is due to the net force of propellers 1 and 3. The net rotation about the z axis is due to the net torques induced by each propeller about their own central axes. The signs for the terms in equation (18) are due to the clockwise rotation of propellers 2 and 4, and the anticlockwise rotation of propellers 1 and 3.

$$l = r(f_4 - f_2) \tag{16}$$

$$m = r(f_3 - f_1) \tag{17}$$

$$n = -\tau_{M1} + \tau_{M2} - \tau_{M3} + \tau_{M4} \quad . \tag{18}$$

A Simulink implementation of the above equations is shown in Figure 16.

Control Allocation

The purpose of control allocation is to map the controller's output (Z_T, l, m, n) to propeller angular velocities required to induce those values.

Equations (15), (16), (17), and (18) can be written as the following matrix equation:

$$\begin{bmatrix} k & k & k & k \\ 0 & -rk & 0 & rk \\ -rk & 0 & rk & 0 \\ -b & b & -b & b \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} = \begin{bmatrix} Z_T \\ l \\ m \\ n \end{bmatrix} .$$

By left-multiplying each side of this equation by the inverse of the matrix on the left-hand side, we can calculate the squared angular velocities of each propeller:

$$\begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} = \begin{bmatrix} k & k & k & k \\ 0 & -rk & 0 & rk \\ -rk & 0 & rk & 0 \\ -b & b & -b & b \end{bmatrix}^{-1} \begin{bmatrix} Z_T \\ l \\ m \\ n \end{bmatrix} . \tag{19}$$

The parameters of the matrix are all known, and given in Table 1. The values of Z_T, l, m, and n are output by the controller block. The numerical inverse of the matrix above is:

$$\begin{bmatrix} k & k & k & k \\ 0 & -rk & 0 & rk \\ -rk & 0 & rk & 0 \\ -b & b & -b & b \end{bmatrix}^{-1} = 10^6 \begin{bmatrix} 0.0839 & 0 & -0.7457 & -2.193 \\ 0.0839 & -0.7457 & 0 & 2.193 \\ 0.0839 & 0 & 0.7457 & -2.193 \\ 0.0839 & 0.7457 & 0 & 2.193 \end{bmatrix} .$$

A Simulink implementation of equation 19 can be seen in Figure 17. Note that the block outputs the squared angular velocities. The Physical Forces and Moments block in the implementation uses only the squared angular velocities, so the square root was not taken here in order to reduce computational complexity.

Controller

The system has four reference inputs: desired height, and 3 Euler angles representing desired orientation. The controller takes the error in these four values and outputs values for Z_T, l, m, and n which would stabilize the system. These values go through the control allocation block, where they are converted into propeller angular velocities.

To simplify the creation of the controller, it was assumed that the single system of four inputs and four outputs was actually four systems of one input and one output each:

- $h \to Z_T$,
- $\phi \to l$,
- $\theta \to m$,
- $\psi \to n$.

In designing each of the four controllers, it was assumed that the inputs to the other three controllers was zero. This yields controllers which are not ideal when multiple inputs are given at once, but which do stabilize the system for one step input at a time. The reference step inputs given were:

- $h_r = 1$ m
- $\phi_r = 0.2$ rad
- $\theta_r = 0.2$ rad
- $\psi_r = 0.2$ rad

As a measure of controller performance, we introduce the following scoring metric:

$$T_s = T_\phi + T_\theta + T_\psi + 0.01T_h, \tag{20}$$

where each T term is the time taken until the system response settles into the region $\pm 2\%$ of the reference input value for the variable in the subscript.

Simulink's built-in PID Controller block was used for each of the four controllers. This block represents a transfer function of the form:

$$P + I\frac{1}{s} + D\frac{N}{1 + N\frac{1}{s}} \quad .$$

Simulink's built-in PID Controller block includes an automated tuning function using the PID Tuner App. However, due to the nonlinearity of the system, the automatic tuning functionality was imperfect, and for each controller did not succeed in stabilizing the actual system. However, the PID Tuner App was still used to find stable gains for the controllers.

The automatic tuner is a function of just two parameters: "Response Time" and "Transient Behavior." Adjusting these two settings automatically adjusts the P, I, and D gains appropriately, reducing the time to tune the controller. For each controller, the tuning process followed four steps:

1. By trial-and-error, find a pair of values for Response Time and Transient Behavior such that the system's response does not go to infinity.

2. By trial-and-error, adjust the Response Time value until the settling time in the response cannot be minimized further.

3. By trial-and-error, adjust the Transient Behavior value until the settling time in the response cannot be minimized further.

4. Export the P, I, D and N values from the tuner to the PID controller block.

The best values for each of the four controllers' gains are listed in Table 2. The controller block from the Simulink implementation is shown in Figure 18. The complete Simulink diagram, with the inputs, controller, control allocation, time delay, physical forces and moments, 6-DOF model, and feedback paths is shown in Figure 19.

The system response plots for each of the four outputs are shown in Figures 20 through 23.

Results

Each output response is considered "settled" at time t_s such that for the given reference value r, the response never exceeds $r \pm 0.02r$. For the height, with a reference of 1 m, the bounds are 1.02 m and 0.98 m. For the Euler angles, each with references of 0.2 rad, the bounds are 2.04 rad and 1.96 rad.

It was visually determined that each response was successfully settled by checking that the value had an error approaching zero within the first 100 seconds of response (seen in Figures 20 through 23). The numerical arrays were exported to MatLab, from which it could be determined the final time each value was out of the settling boundaries.

The settling times for each of the four values were:

- $T_h \approx 37.34$ seconds
- $T_\phi \approx 4.07$ seconds
- $T_\theta \approx 4.04$ seconds

- $T_\psi \approx 4.80$ seconds

Calculating the score from equation (20):

$$\begin{aligned} T_s &= 4.07 + 4.04 + 4.80 + 0.01 \cdot 37.34 \\ &= \boxed{13.28} \quad . \end{aligned}$$

Conclusion

We have derived the 6-DOF equations of motion for a quadcopter, the relation between the angular velocities of the quadcopter propellers and the forces and moments induced on the vehicle, and we have developed a controller to stabilize the system for a desired height and orientation.

We calculated the settling time score metric defined in equation (20) to be $\boxed{13.28}$ seconds.

Tables and Figures

Variable	**Description**	**Value**	**Unit**
g	Gravitational Acceleration	9.81	$\frac{m}{s^2}$
m	Quadcopter Mass	0.468	kg
l	Quadcopter Arm Length	0.225	m
k	Propeller Lift Coefficient	$2.980 \cdot 10^{-6}$	–
b	Propeller Drag Coefficient	$1.140 \cdot 10^{-7}$	–
I_{xx}	Inertia Matrix R1C1	$4.856 \cdot 10^{-3}$	$kg \cdot m^2$
I_{yy}	Inertia Matrix R2C2	$4.856 \cdot 10^{-3}$	$kg \cdot m^2$
I_{zz}	Inertia Matrix R3C3	$8.801 \cdot 10^{-3}$	$kg \cdot m^2$
U	Inertial Frame Force x Component	Variable	N
V	Inertial Frame Force y Component	Variable	N
W	Inertial Frame Force z Component	Variable	N
ϕ	Inertial Frame Euler Angle About x Axis	Variable	rad
θ	Inertial Frame Euler Angle About y Axis	Variable	rad
ψ	Inertial Frame Euler Angle About z Axis	Variable	rad
P	Inertial Frame Moment About x Axis	Variable	N·m
Q	Inertial Frame Moment About y Axis	Variable	N·m
R	Inertial Frame Moment About z Axis	Variable	N·m
h	Quadcopter Height in Inertial Frame	Variable	m

Table 1: Variable Descriptions and Values

Controller	**P**	**I**	**D**	**N**
h	0.66709075982305	0.083340954486102	1.3246835477396	326.52227886903
ϕ	0.00679742603344	0.000126528756853	0.0097998134935	11.93198682033
θ	0.00679742603344	0.000126528756853	0.0097998134935	11.93198682033
ψ	0.00893864028383	0.001100306041983	0.0157373637319	6.9488800045551

Table 2: PID Controller Parameters for Simulink

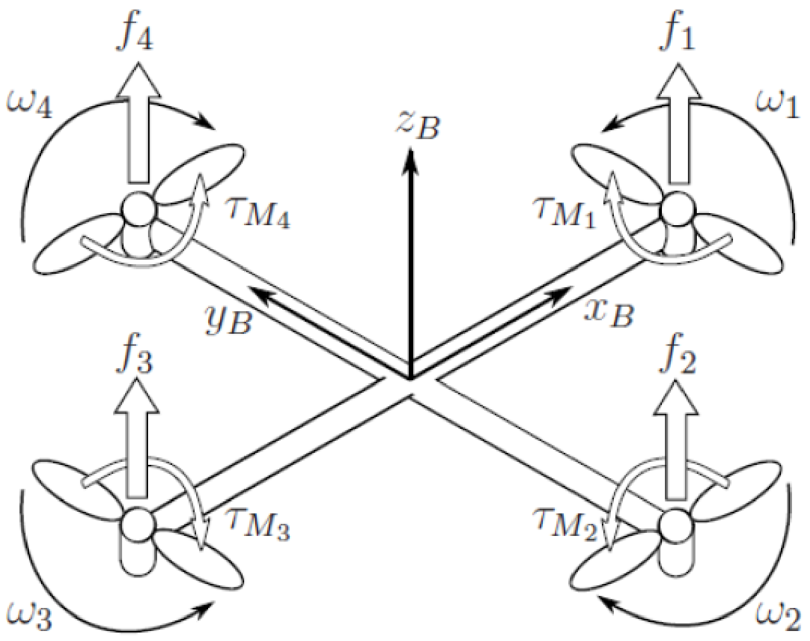

Figure 1: A Quadcopter and Body Frame Axes

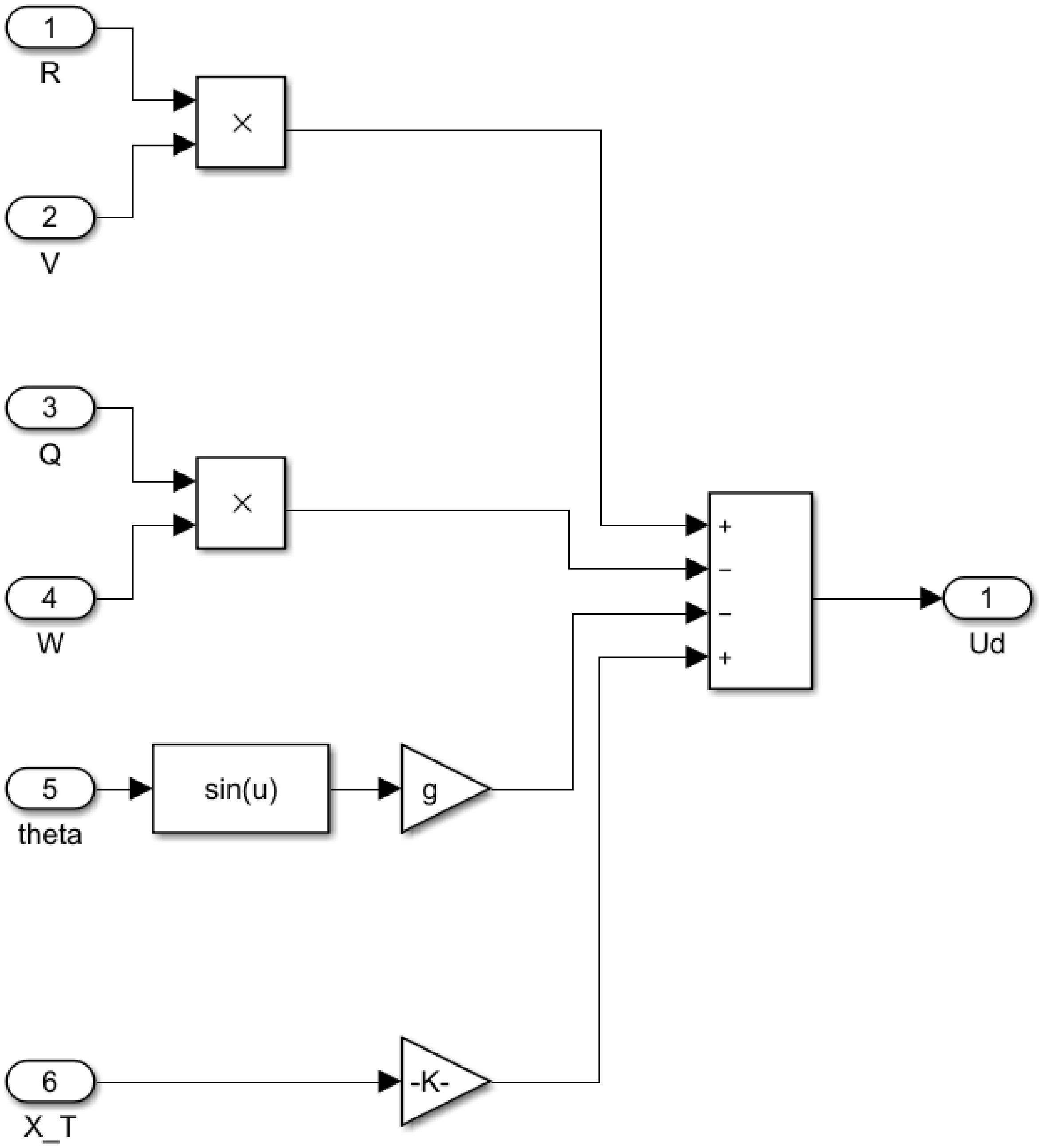

Figure 2: $\dot{U}$ Equation (2) Simulink Implementation

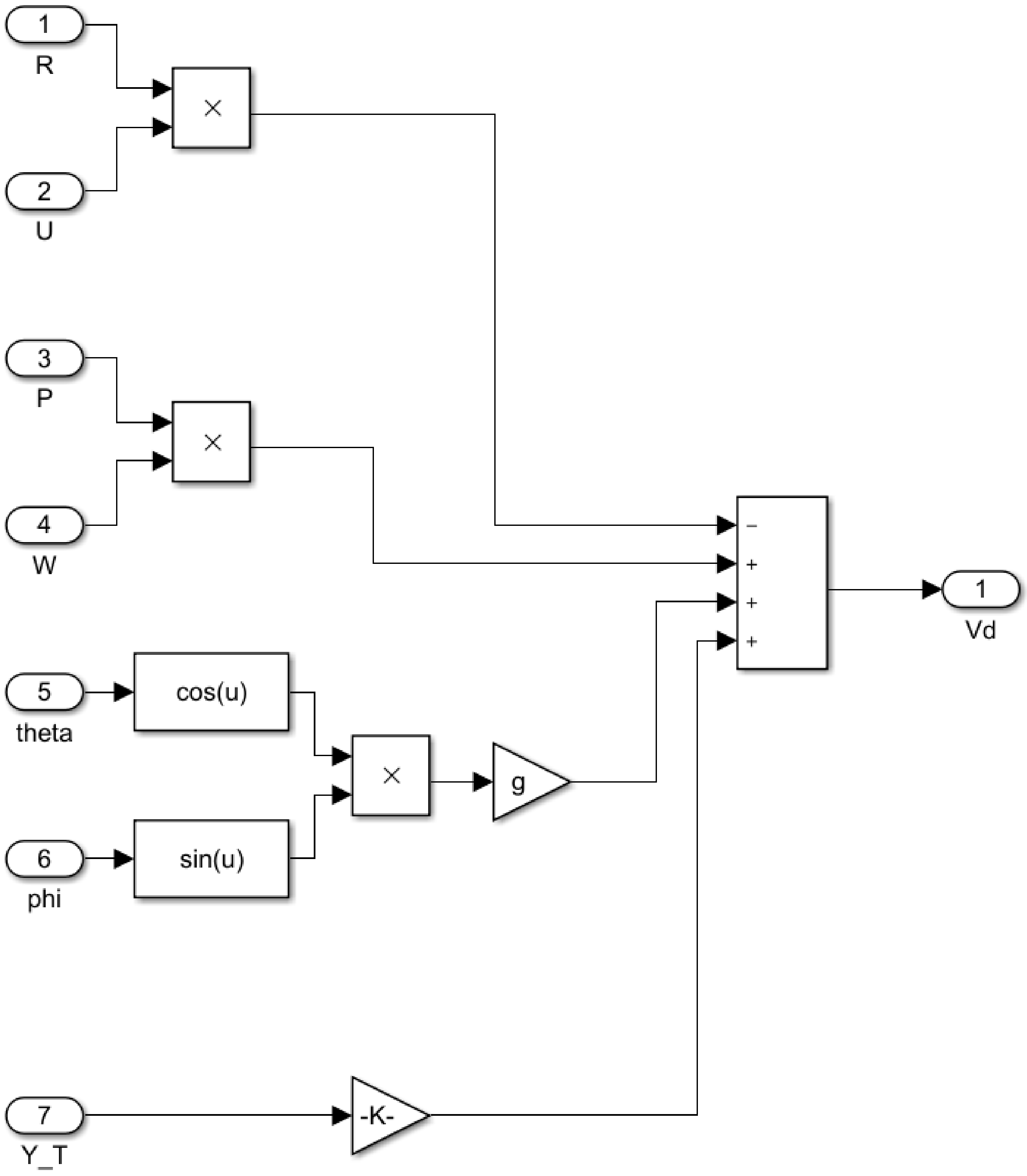

Figure 3: $\dot{V}$ Equation (3) Simulink Implementation

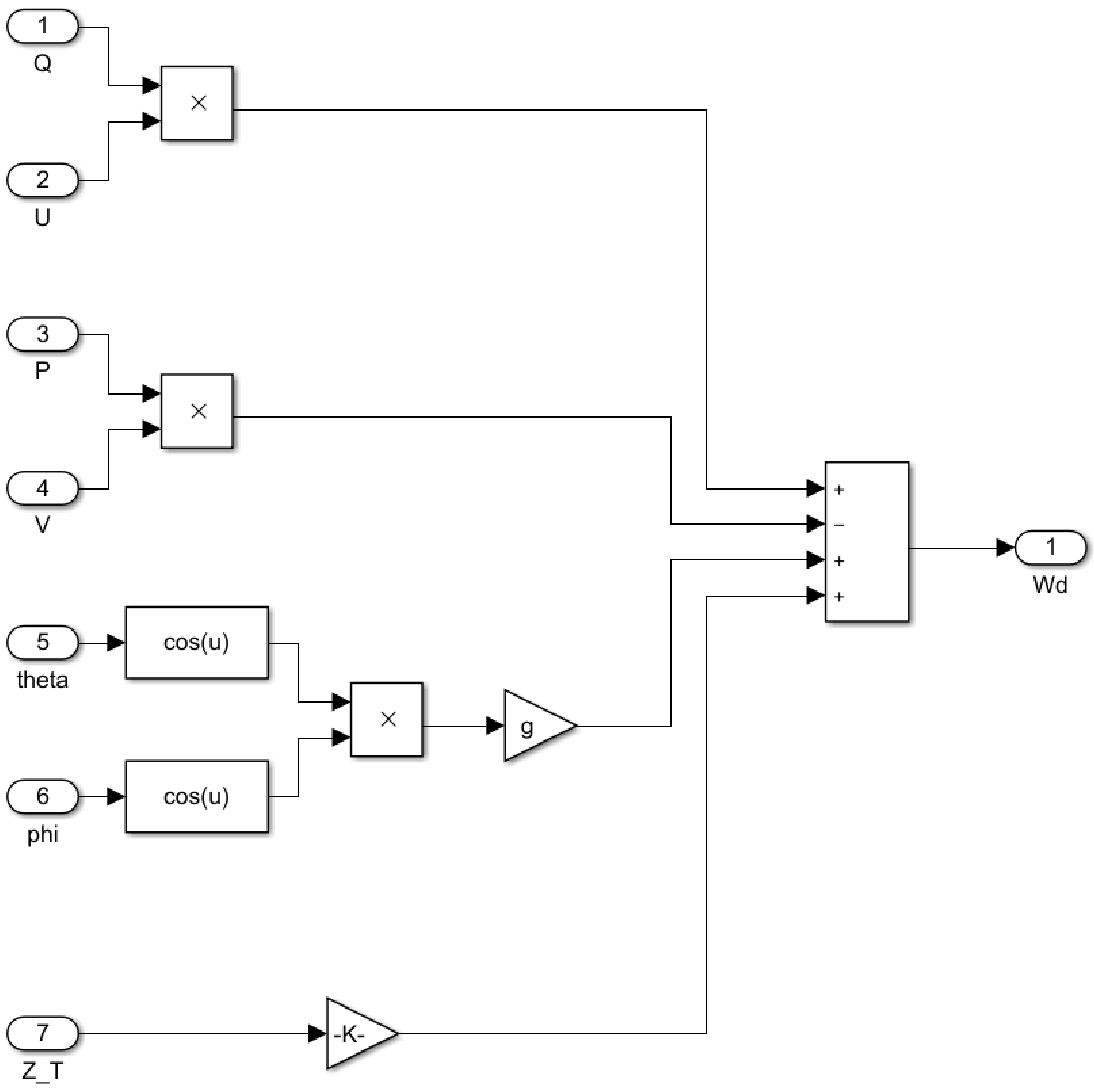

Figure 4: $\dot{W}$ Equation (4) Simulink Implementation

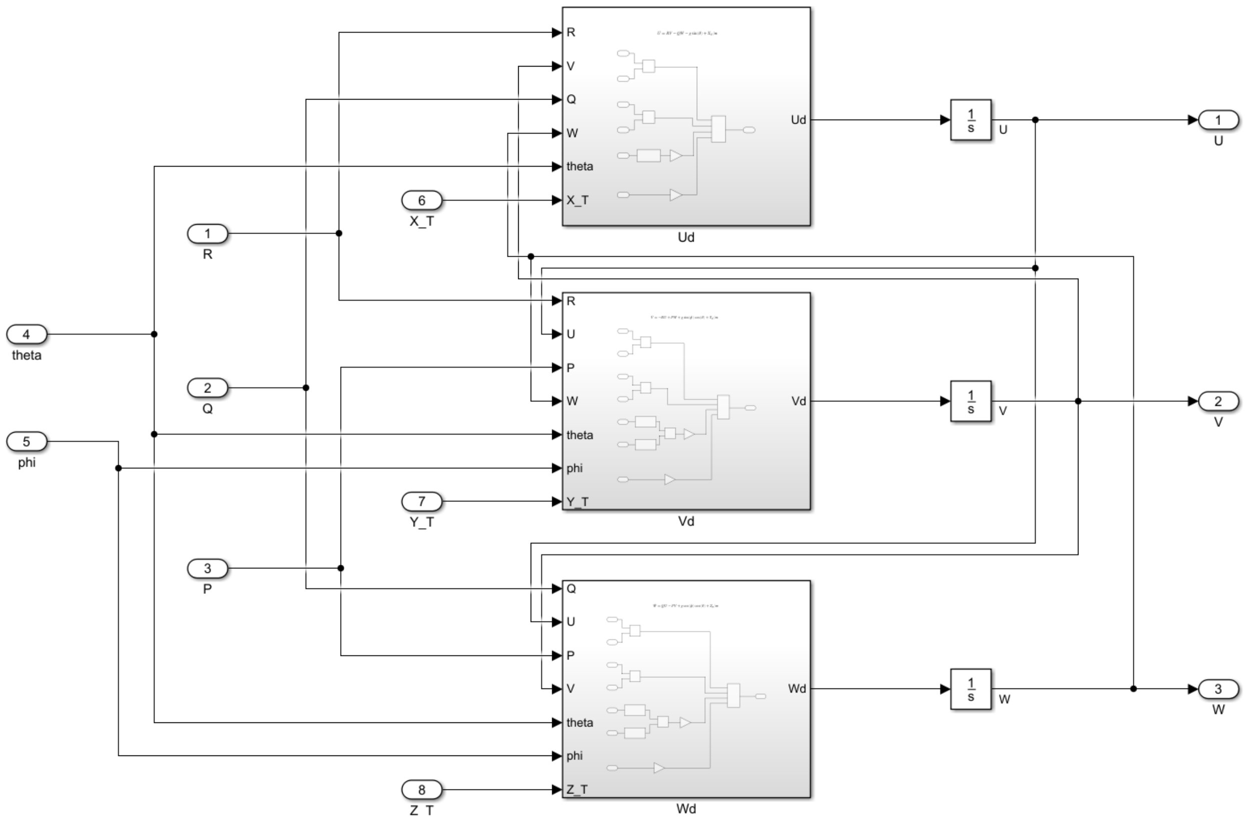

Figure 5: All 3 Force Equation Subsystems Connected in Simulink

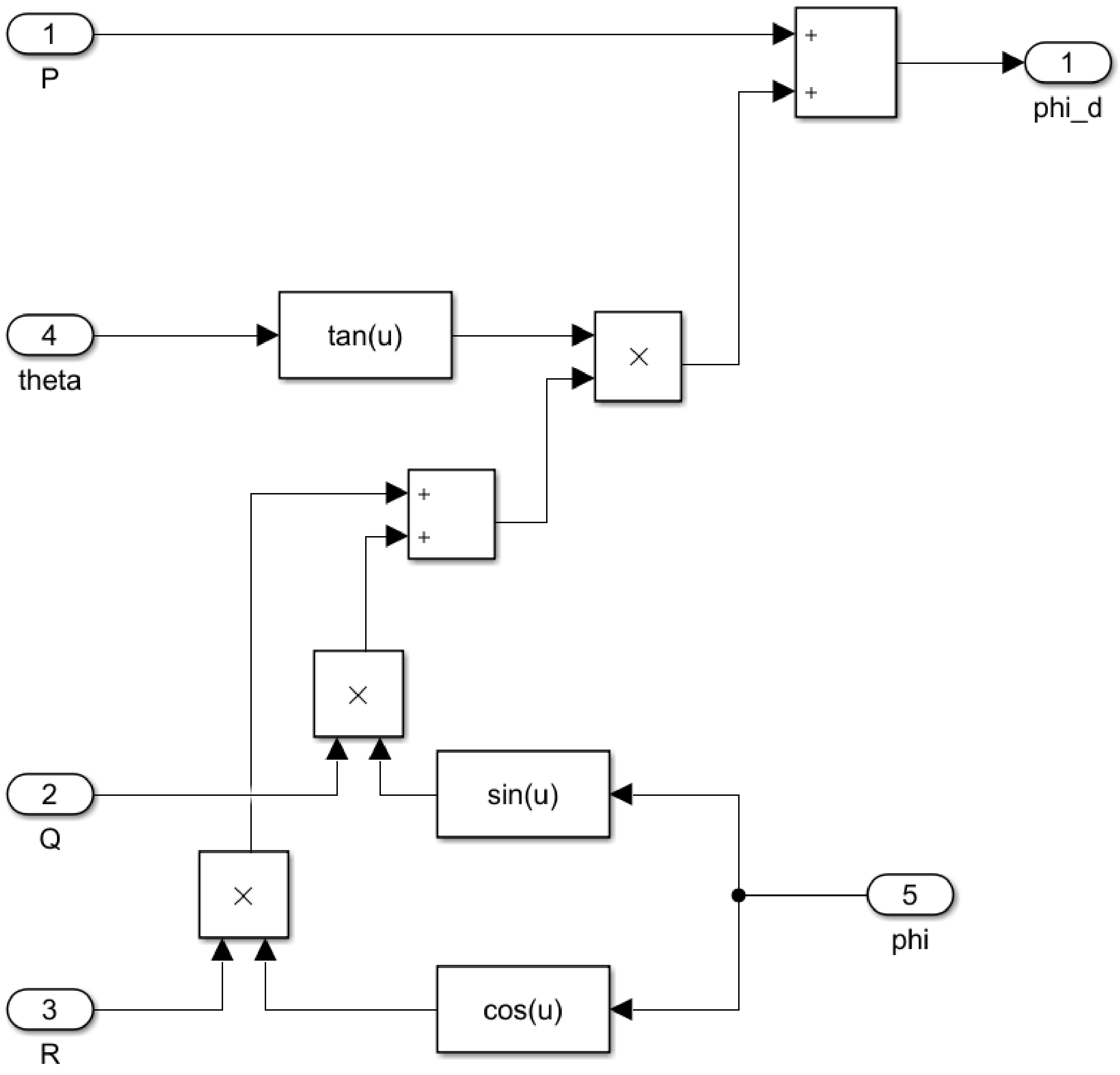

Figure 6: $\dot{\phi}$ Equation (5) Simulink Implementation

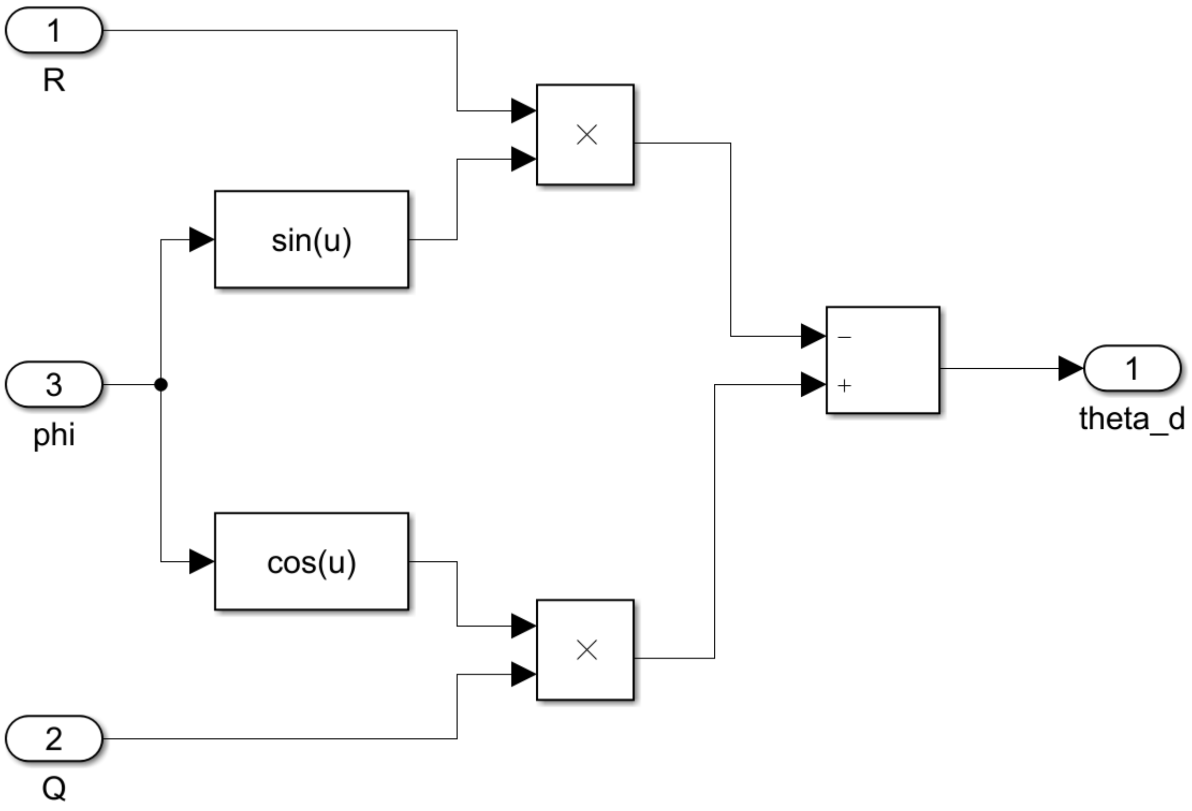

Figure 7: $\dot{\theta}$ Equation (6) Simulink Implementation

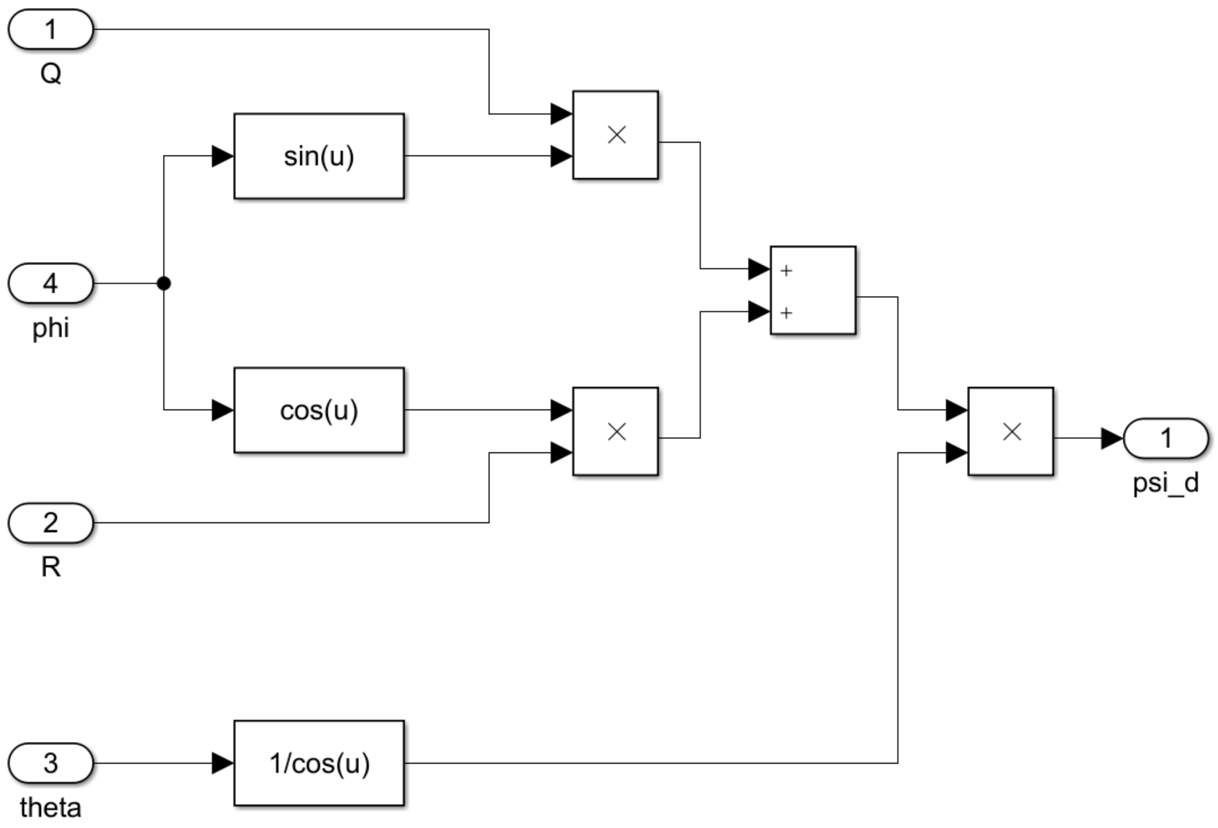

Figure 8: $\dot{\psi}$ Equation (7) Simulink Implementation

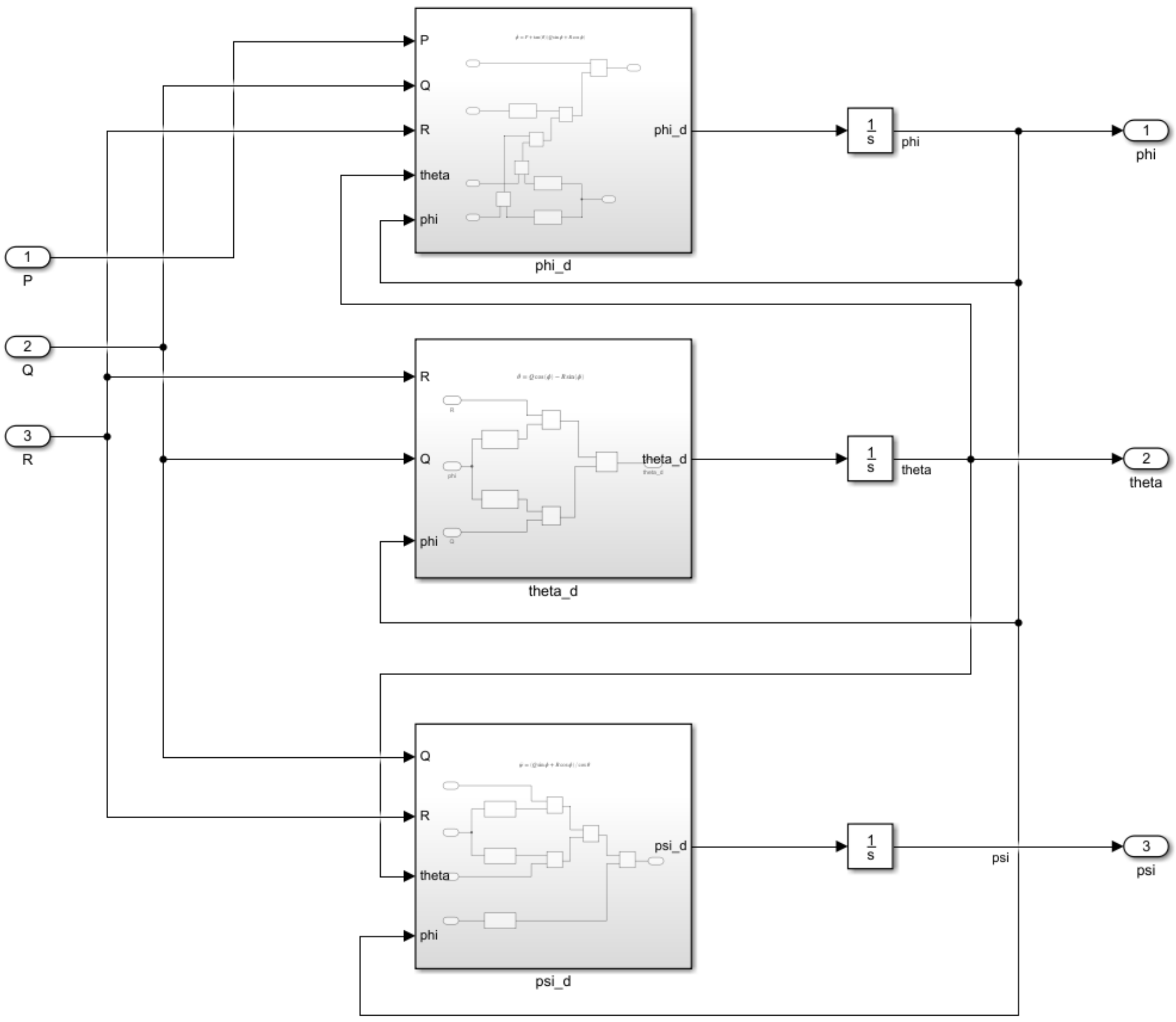

Figure 9: All 3 Kinematic Equation Subsystems Connected in Simulink

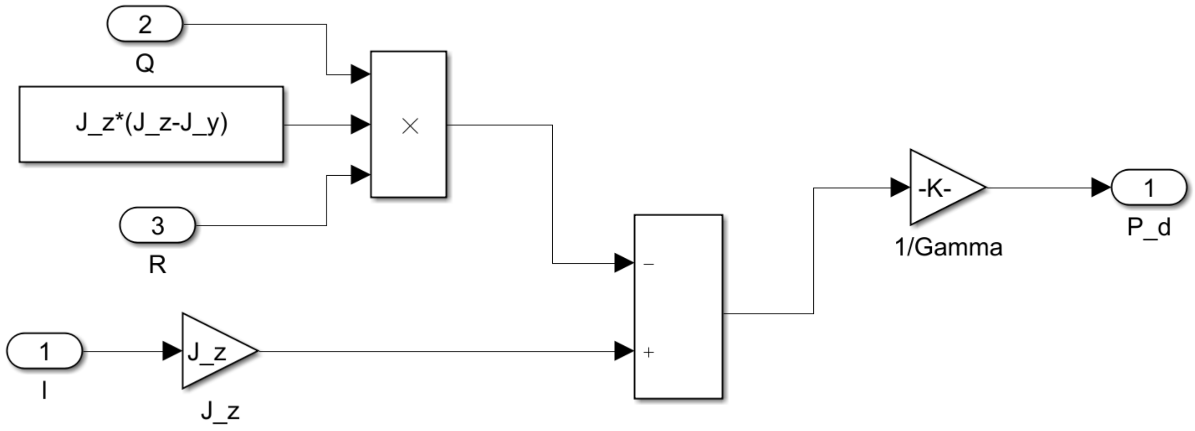

Figure 10: $\dot{P}$ Equation (8) Simulink Implementation

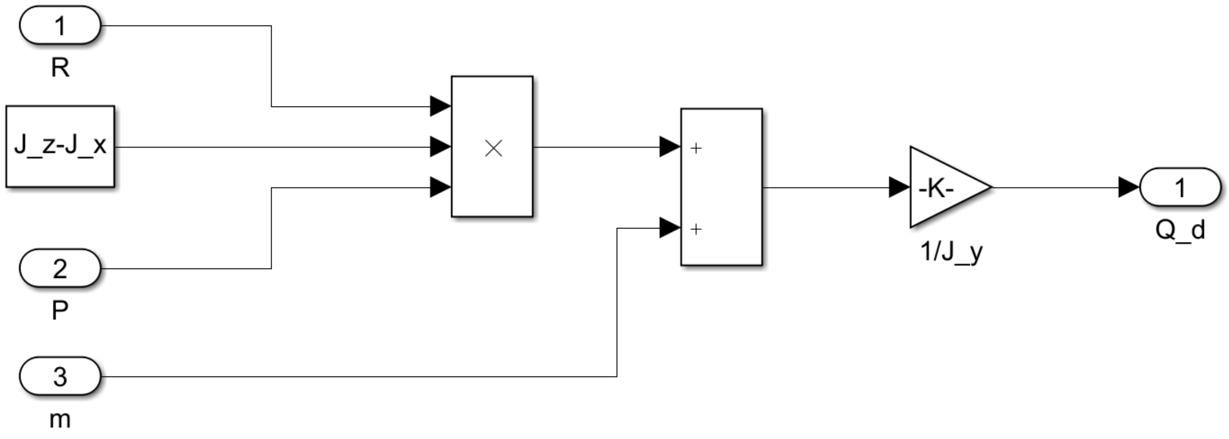

Figure 11: $\dot{Q}$ Equation (9) Simulink Implementation

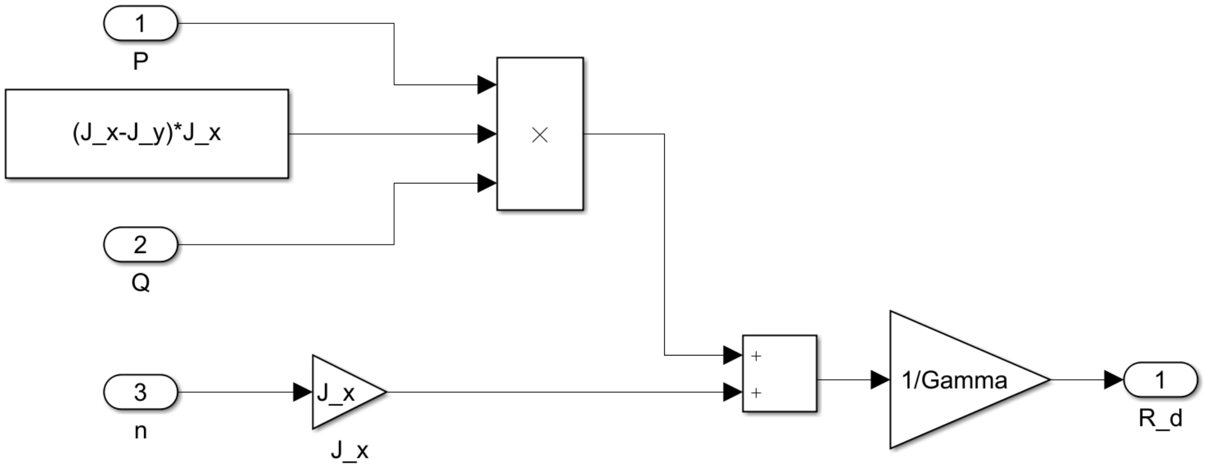

Figure 12: $\dot{R}$ Equation (10) Simulink Implementation

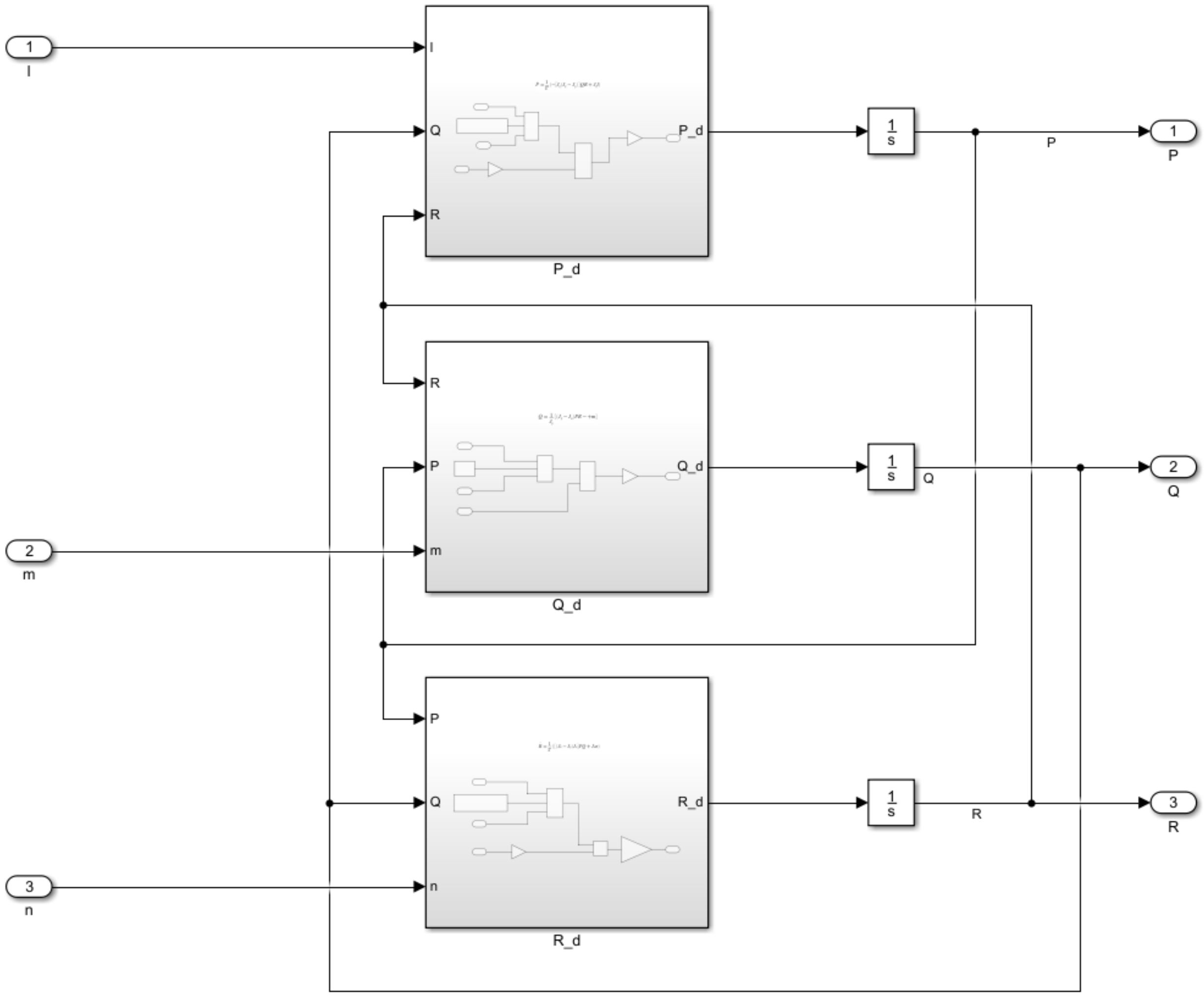

Figure 13: All 3 Moment Equation Subsystems Connected in Simulink

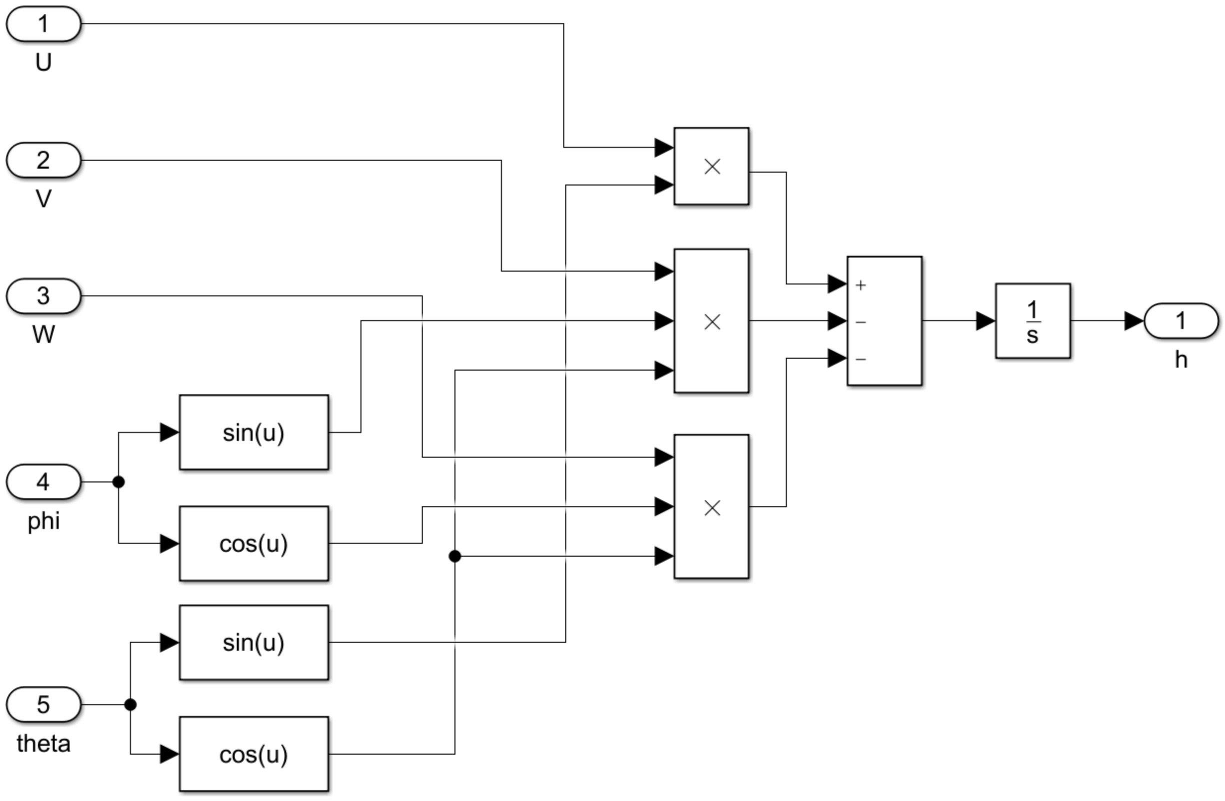

Figure 14: $\dot{h}$ Equation (11) Simulink Implementation

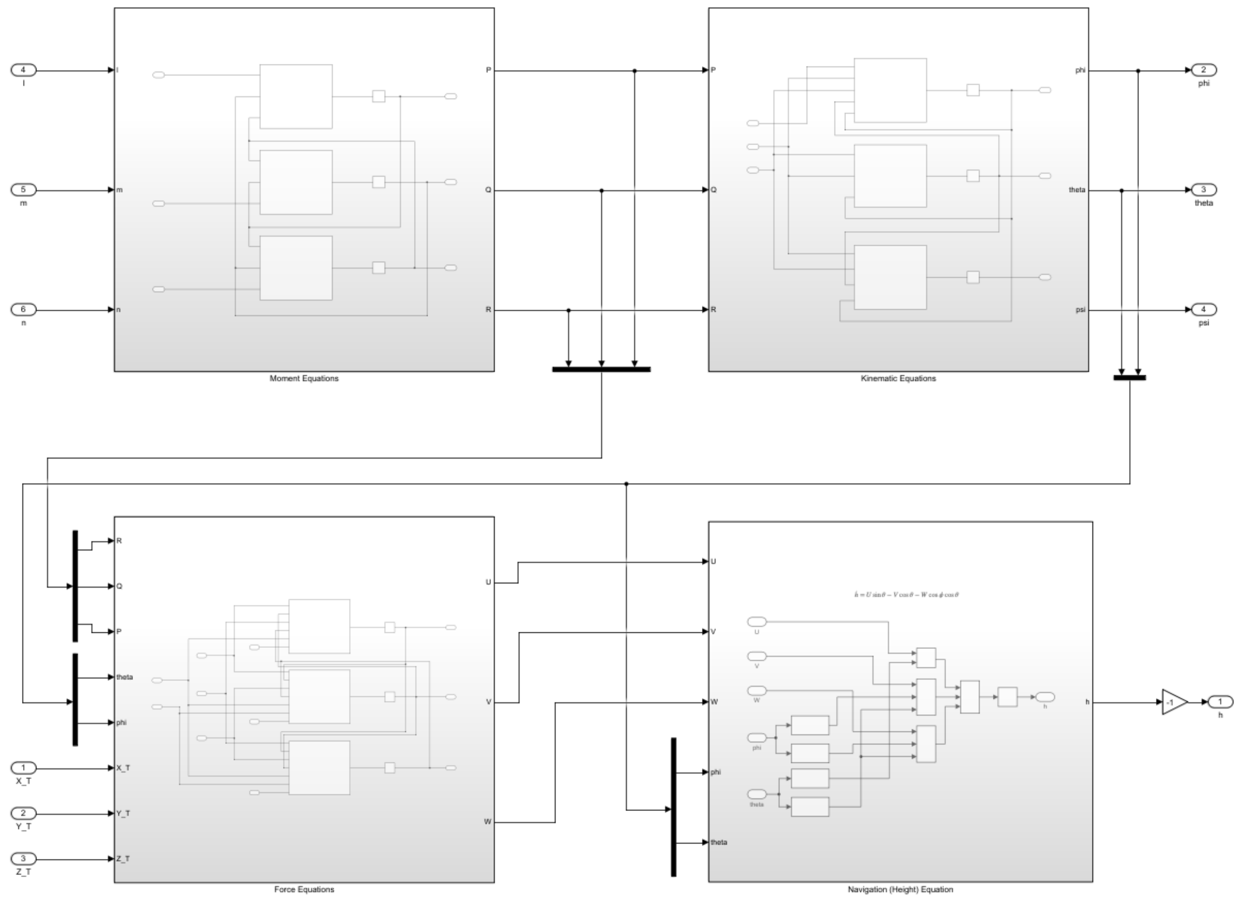

Figure 15: All 6-DOF Model Subsystems Connected in Simulink

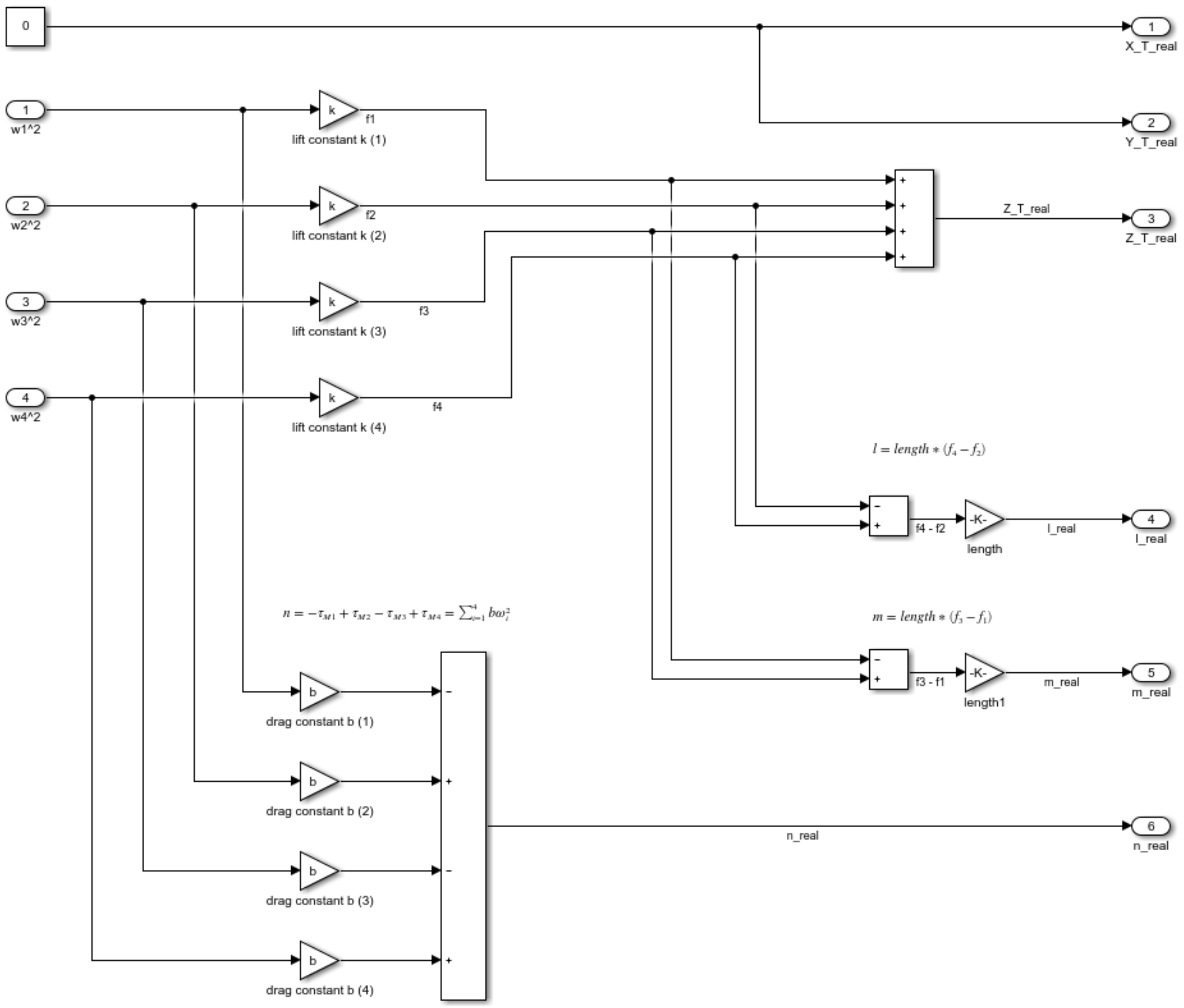

Figure 16: Physical Forces and Moments Equations in Simulink

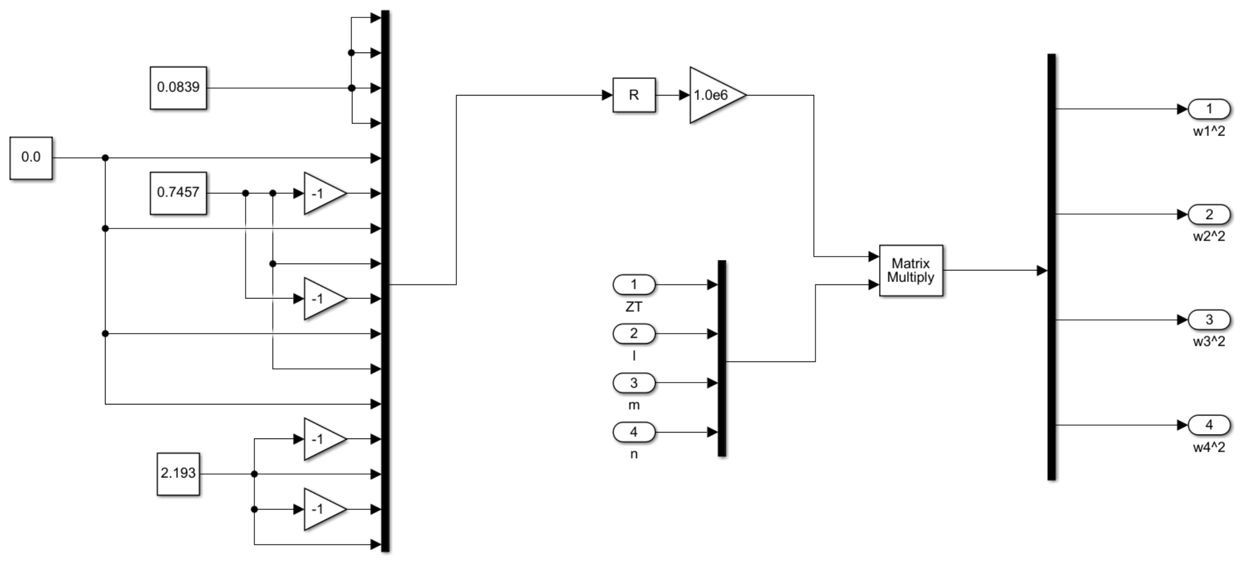

Figure 17: Controller Allocation in Simulink

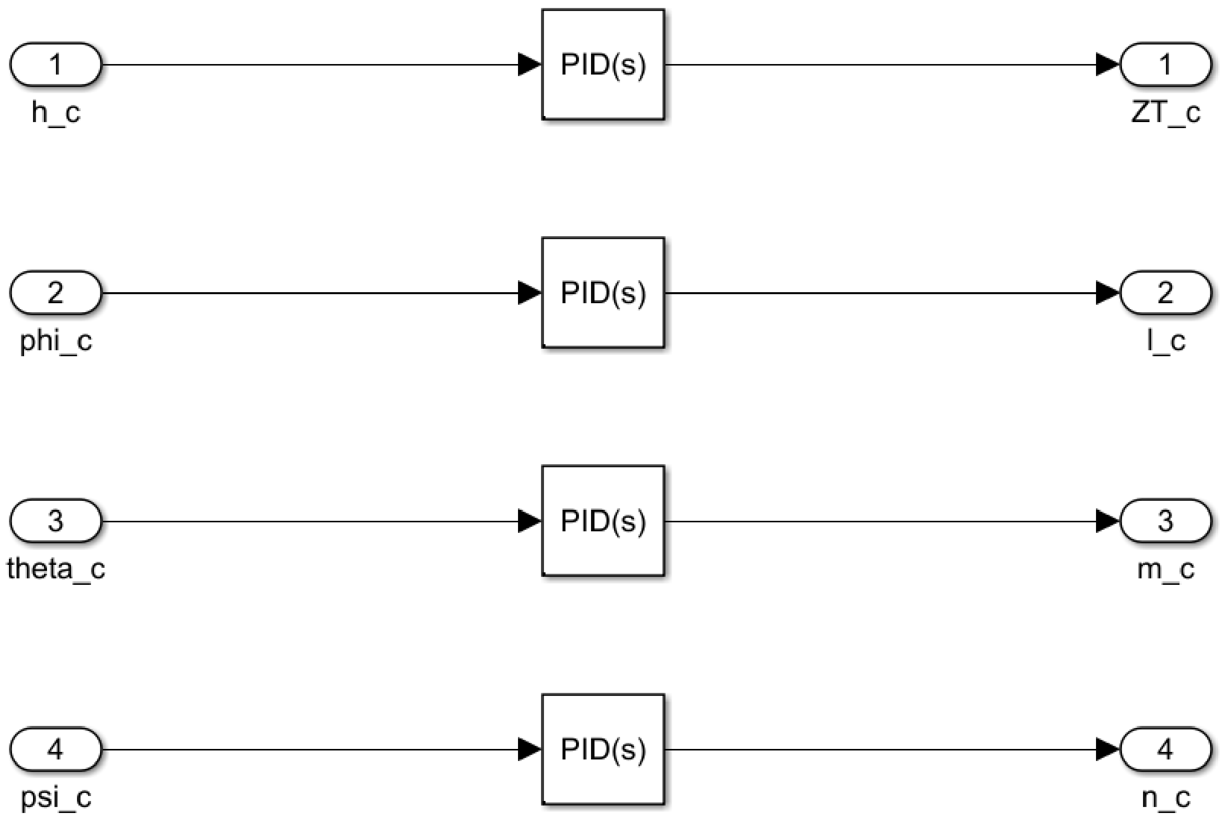

Figure 18: PID Controllers in Simulink

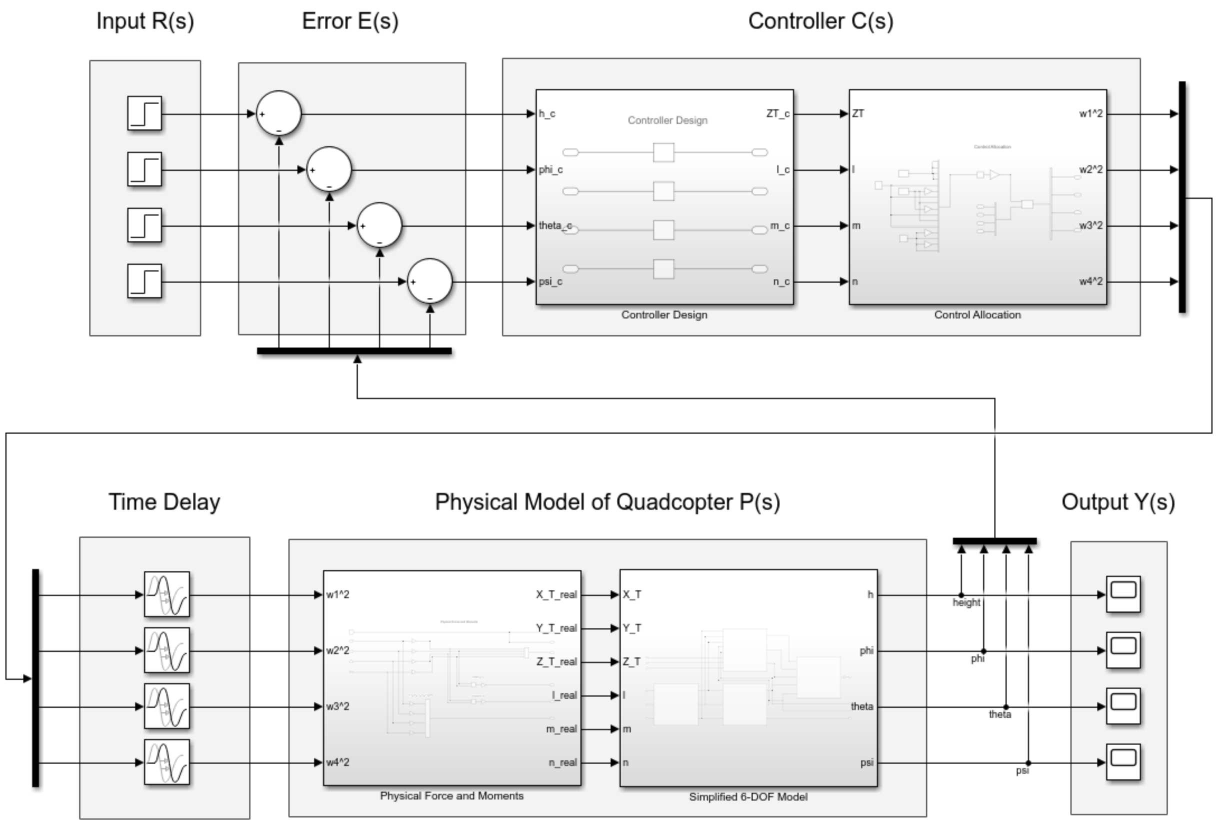

Figure 19: The Entire System Represented in Simulink

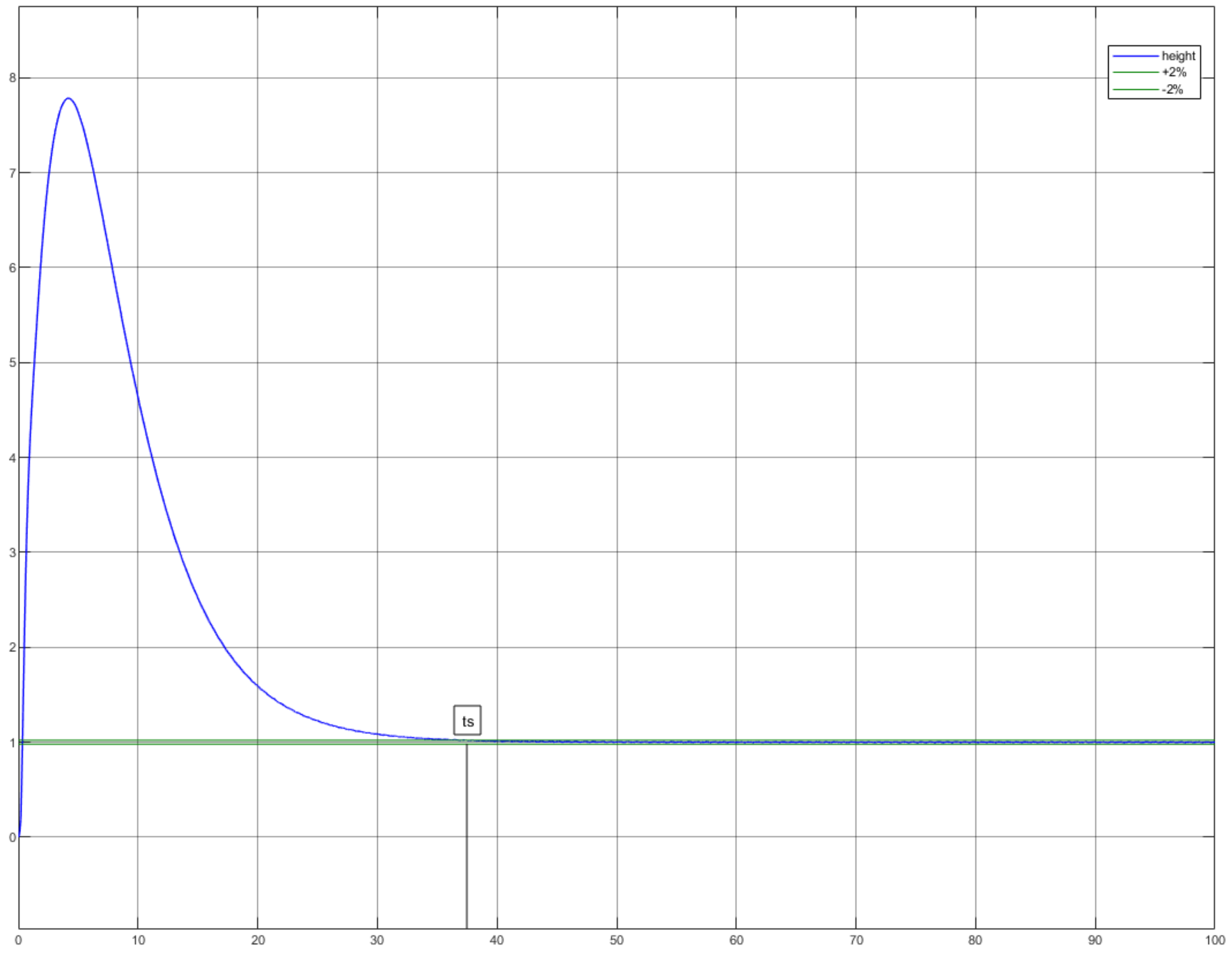

Figure 20: Height Response Plot

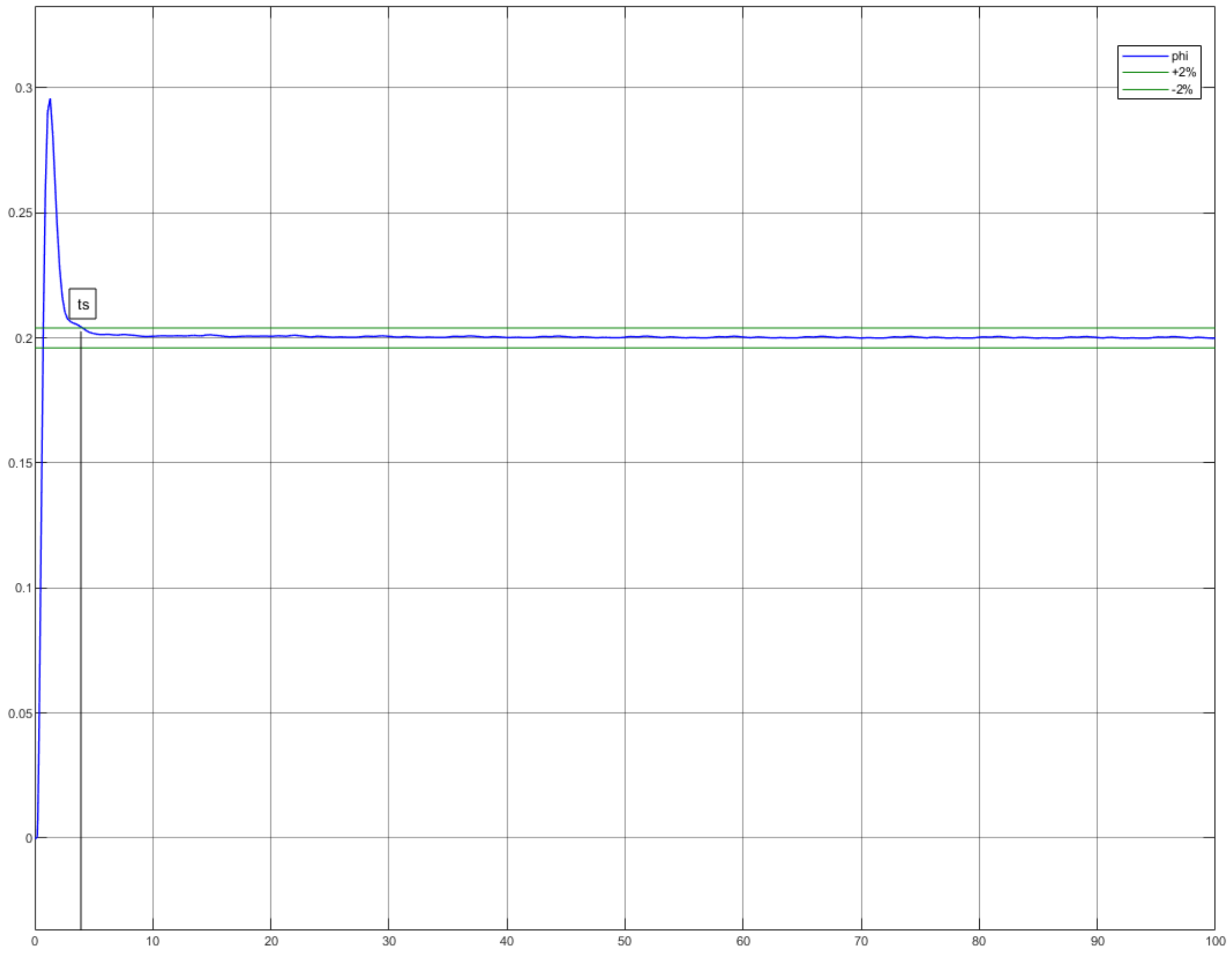

Figure 21: ϕ Response Plot

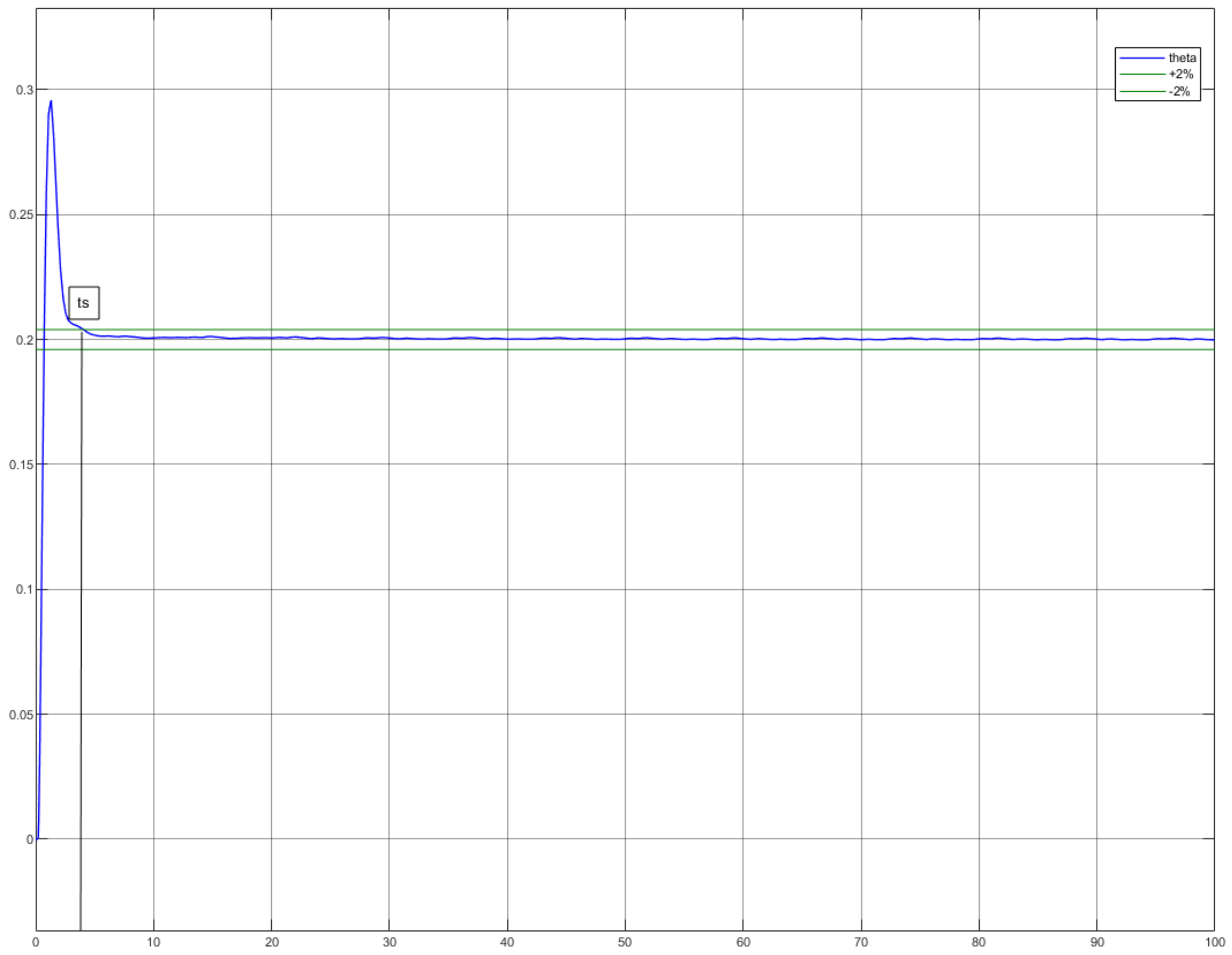

Figure 22: θ Response Plot

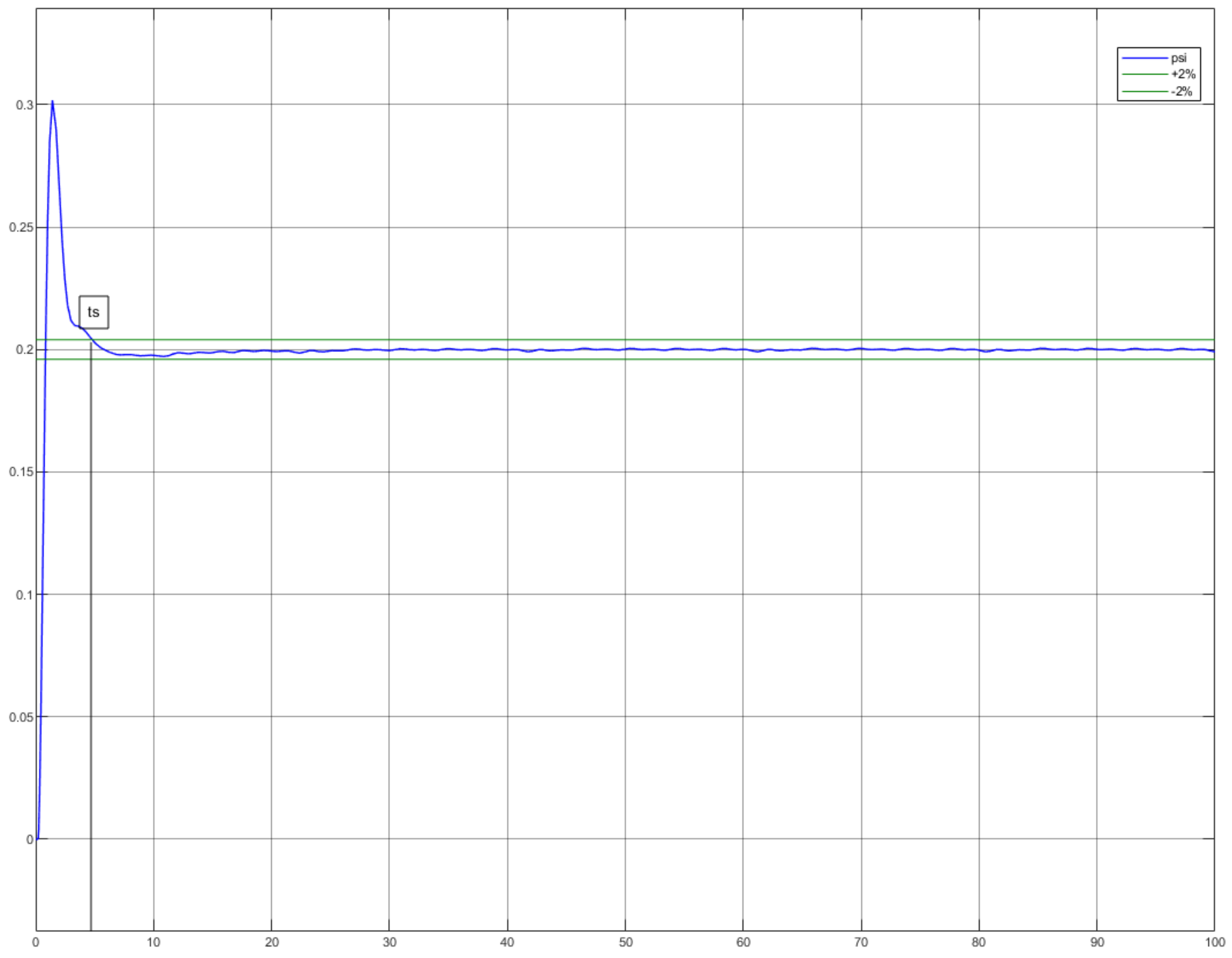

Figure 23: ψ Response Plot