



From Spring Mass Damper Dynamics to Drone Control Models

Prathyush R. Bangalore

Abstract

Complex dynamical systems can be modeled clearly by building them from simpler systems with similar mathematical structure. We begin with a one-dimensional spring-mass-damper system to establish a basic state-space framework, then extend this framework to three-dimensional translational motion and quaternion-based rotational motion. The 3D model is finally combined with rotor dynamics into a unified first-order nonlinear drone model. Numerical simulations are used to compare the behavior of the intermediate models and the final drone model. Our results show that simpler models provide a useful foundation for constructing and interpreting more complex systems, while also showing where that foundation must be extended to account for nonlinear orientation dynamics and input-driven control.

1 Introduction

Modeling complex systems such as aircraft flight control, autonomous drones, spacecraft attitude control, robotic manipulators, and computer graphics can be difficult due to non-linear, and three-dimensional. We show that this complexity can be made more manageable by building the model incrementally from simpler systems. Starting from a one-dimensional spring-mass-damper, we gradually extend the model to take into account three-dimensional translation and rotational motion using consternation.

For a drone model we identify three key sources of complexity: translational dynamics, rotational dynamics, and control inputs. To make this complexity manageable, we develop each component using one-dimensional elements before combining them into a full three-dimensional system. We begin with a basic spring-mass-damper(SMD) system to establish a state-space, extend to three-dimensional components and finally arrive at a drone model driven by rotor forces.

State-space modeling represents a system as set of first order differential equations that describe how state variables evolve over time. The smallest set of variables that determine all properties of a system at a given time, evolve over time [1, p. 121]. This makes state-space models useful for comparing systems that may look physically different, but contain mathematically similar state variables. For example, Nise [1, p. 84] shows that a spring-mass-damper system can be treated analogously to a capacitor-inductor circuit, since both contain similarly related state variables, and related differential-equation models. We will expand on this process of using an analogous SMD system to develop a drone model step by step.

Due to the inherent complexity of drones, additional mathematical tools are required in its modeling. Quaternions provide an elegant way to deal with rotation in three-dimensions while avoiding the singularities and gimbal lock issues caused by Euler angles. Quaternions are easily translated to and from Euler angles, and Euler-based angular velocity can be converted quaternion rotation [2, p. 22]. Kumar et al. demonstrate the use of quaternions directly in quadcopter modeling by representing vehicle attitude through quaternion kinematics and using quaternions to transform motion between the body frame and the inertial frame [3]. This supports the use of quaternions in the present paper, where they provide the mathematical link between three-dimensional rotation and the full drone system model.

The main contribution of this paper is to show how a complex drone system can be developed step by step from simpler mathematical models, making the final system easier to understand and simulate. By progressing from a simple spring mass damper model to a complex quadcopter model, we aim to give a concrete example of how this progression can be practically done and encourage others to follow this approach in modeling other complex systems.

2 Problem Statement

Modeling a drone requires describing how position, velocity, orientation and angular velocity evolve over time under applied forces and torques. These quantities are coupled and nonlinear, making the full system difficult to build and interpret directly.

A drone cannot be modeled by considering each propeller force independently. The motion of the drone depends on the combined effect of all propeller forces together with gravity. These forces determine the drone's translational motion, while differences between propeller forces create torques that rotate the drone. Although Newton's second law describes the acceleration caused by a force or torque, the full drone model is more difficult because these effects occur simultaneously on all axes.

The main difficulty is that translation and rotation are coupled. When a torque changes the drone's orientation, it also changes the direction in which the propellers apply thrust relative to the world frame. This means that rotation directly affects translational motion, and the resulting motion can also affect later forces and torques. As a result, the drone's position, velocity, orientation, and angular velocity cannot be treated as separate quantities.

To make this complexity more manageable, we first use a spring-mass-damper system as a structural analogy. A spring-mass-damper (SMD) system allows us to study how forces change the motion of a mass along one axis before introducing rotation, multiple inputs, and three-dimensional coupling. This simpler model does not capture the full behavior of a drone, but it provides a clear starting point for developing the state-space framework that is later extended into the complete drone model.

2.1 One-Dimensional SMD Model

A spring-mass-damper system consists of a mass m attached to a spring with stiffness k and a damper with coefficient c . When the mass is displaced from its rest position, the spring pulls it back while damper resists its motion. The displacement x of the mass from its rest position fully describes the configuration of the system, as show in Figure 1

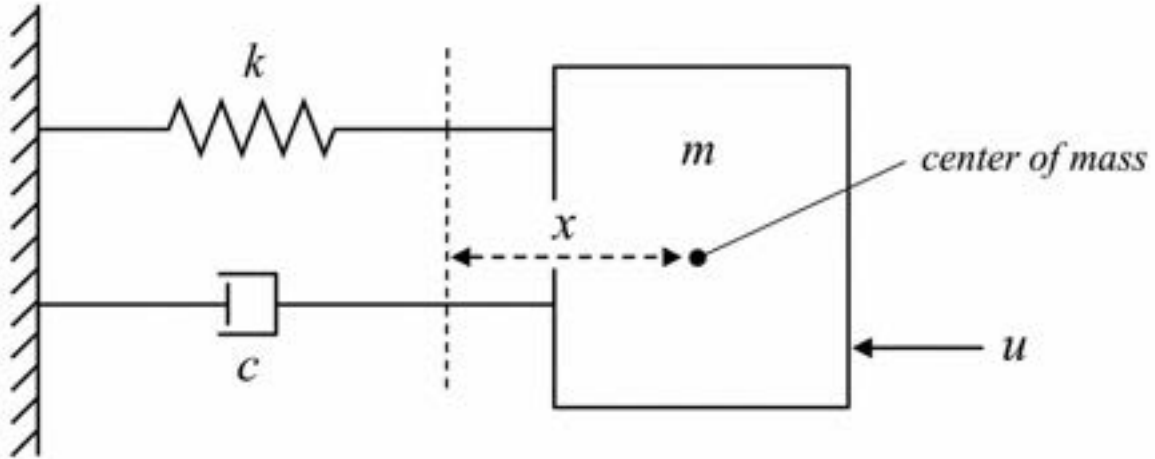


Figure 1

The state of the system is described by its position and velocity. Knowing both tells us everything about how the system will behave going forward. We therefore define the state vector as:

$$\mathbf{x} = \begin{bmatrix} x \\ \dot{x} \end{bmatrix}$$

To determine how the state evolves over time, we apply Newton's second law to find the net force acting on the mass. The system has two physical components, each contributing one force. The spring exerts a restoring force describe Hooke's law, which is proportional to displacement, The damper exerts a resistive force proportional to velocity. Together these two forces fully determine the acceleration for the mass. These relationships are given by:

$$F_s = -kx \quad (1)$$

$$F_d = -c\dot{x} \quad (2)$$

Applying Newton's second law, the equation of motion of the system is:

$$\begin{aligned} F &= ma \\ F_s + F_d &= ma \\ -kx - c\dot{x} &= m\ddot{x} \\ \ddot{x} &= -\frac{k}{m}x - \frac{c}{m}\dot{x} \end{aligned} \quad (3)$$

Due to Eq. 3 being a second order differential equation, it is difficult to directly evaluate or solve. We can instead express it in state-space form as a set of first order differential equations.

Eq. 3 can be represented in state-space as:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ -\frac{k}{m}x - \frac{c}{m}\dot{x} \end{bmatrix} \mathbf{x}$$
$$\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix}$$

To model a controllable system, we introduce an external input force u , applied directly to the mass. In this section, u is a simple push force along one axis. This concept of u as an external input carries through every model in this paper, taking on a different physical meaning at each stage while always serving as the mechanism through which the system is actively driven

$$u = ma$$
$$\ddot{x} = -\frac{k}{m}x - \frac{c}{m}\dot{x} + \frac{1}{m}u$$

The corresponding state space form is:

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u \quad (4)$$

This allows the system to be influenced by external force u , enabling active control and corrections.

We can see in this representation that the system is linear and time invariant, and so can also be represented as:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u \quad (5)$$
$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix}$$
$$\mathbf{B} = \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix}$$

2.2 Three-Dimensional System Extension

The one-dimensional formulation provides a foundation for understanding system dynamics in terms of force and motion. The state of the mass was described by its position

and velocity along a single axis. We now extend this model to three-dimensions by applying the same dynamics along each spatial axis. However, in three-dimensions, we must also consider rotational dynamics, which introduce nonlinear behavior and coupling between states.

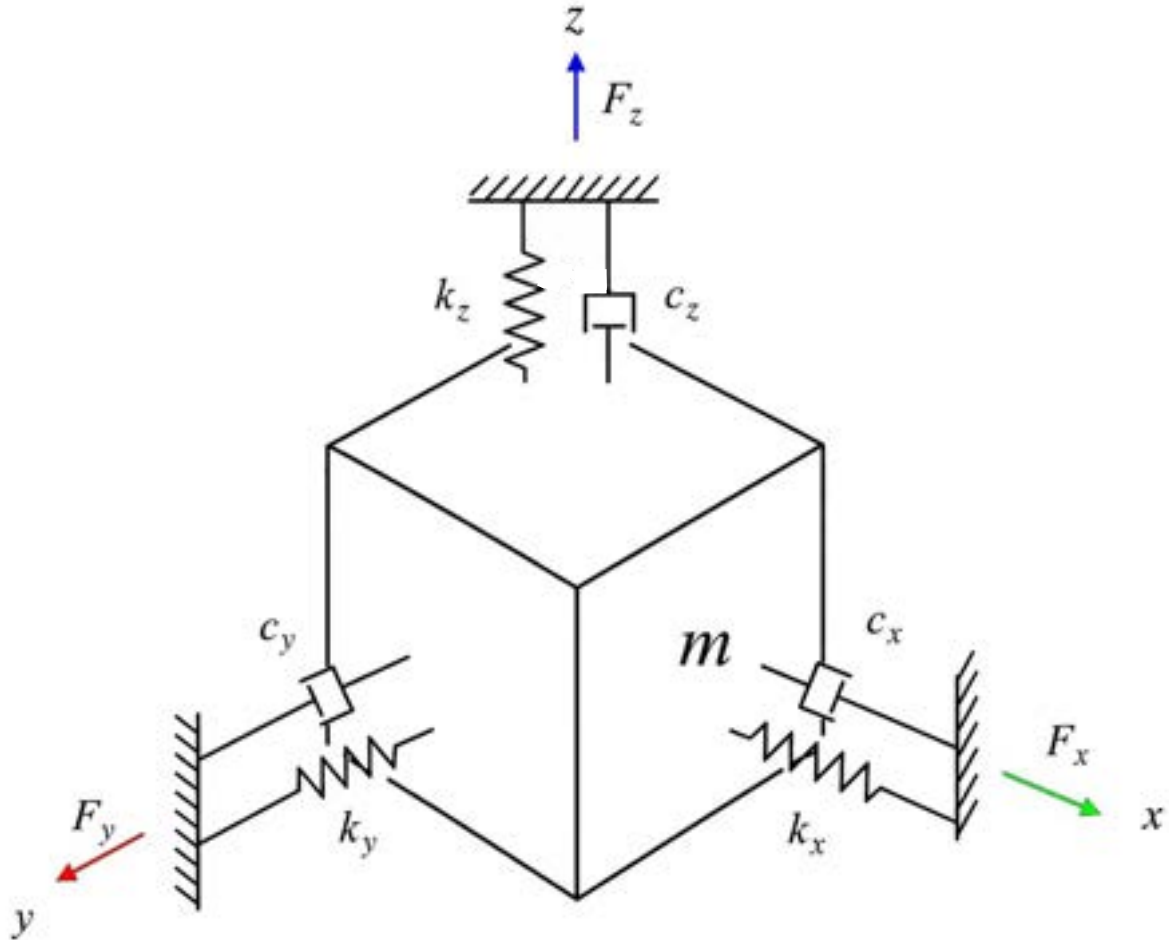


Figure 2

2.2.1 Translational Dynamics

We extend the spring-mass-damper system to three-dimensions by defining the position of the mass as:

$$\mathbf{x}_p = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Where x , y , and z represent the position of the object along each special axis.

The state vector is:

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}$$

Assuming decoupled motion along each axis, the dynamics follow directly from the one-dimensional equation of motion in Eq. 3:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ -\frac{k_x}{m}x - \frac{c_x}{m}\dot{x} \\ -\frac{k_y}{m}y - \frac{c_y}{m}\dot{y} \\ -\frac{k_z}{m}z - \frac{c_z}{m}\dot{z} \end{bmatrix}$$

To model controlled motion, we introduce an external input force, representing the applied force along each spatial axis:

$$\mathbf{u} = \mathbf{F} = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix}$$

Each component acts independently on its axis, directly analogous to the single forces u in the one-dimension, The system remains linear and each axis retains the same passive restoring behavior from its spring.

Since the system is linear, it can be expressed in state-space form in Eq. 6, consistent with the one-dimensional system:

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -\frac{k_x}{m} & 0 & 0 & -\frac{c_x}{m} & 0 & 0 \\ 0 & -\frac{k_y}{m} & 0 & 0 & -\frac{c_y}{m} & 0 \\ 0 & 0 & -\frac{k_z}{m} & 0 & 0 & -\frac{c_z}{m} \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{1}{m} & 0 & 0 \\ 0 & \frac{1}{m} & 0 \\ 0 & 0 & \frac{1}{m} \end{bmatrix}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -\frac{k_x}{m} & 0 & 0 & -\frac{c_x}{m} & 0 & 0 \\ 0 & -\frac{k_y}{m} & 0 & 0 & -\frac{c_y}{m} & 0 \\ 0 & 0 & -\frac{k_z}{m} & 0 & 0 & -\frac{c_z}{m} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{1}{m} & 0 & 0 \\ 0 & \frac{1}{m} & 0 \\ 0 & 0 & \frac{1}{m} \end{bmatrix} \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} \quad (6)$$

The independence between the axes will not hold in the drone model. In the drone, F is no longer a free input. F_x and F_y arise from the drone tilting and rotating the thrust vector, coupling rotation and translation together. This coupling is addressed in Section 2.3.

Rotation cannot be handled by the same axis. Orientation in three dimensions requires a different mathematical representation, which is addressed in Section 2.2.2.

2.2.2 Rotational Dynamics

In the translational system, the state was described by position and velocity along each axis independently. We now extend this idea to rotation. The rotation state is similarly described by two quantities: orientation and angular velocity. Just as velocity drives position forward in time, angular velocity drives orientation forward in time.

However, unlike translation, rotational axes in 3D are not independent. Rotating around one axis changes the meaning of the other axes. Three separate angles cannot represent uniquely in all cases – when two axes align, a degree of freedom is lost. This is the gimbal lock problem with Euler angles.

Quaternions solve this problem by representing orientation differently. Instead of three sequential angle or rotations, a quaternion describes any 3D orientation as a single rotation around one axis in space. To represent rotational motion, we use quaternions to describe orientation in three-dimensional space. A quaternion is defined as:

$$q = \begin{bmatrix} q_w \\ q_x \\ q_y \\ q_z \end{bmatrix}$$

Where q_x, q_y and q_z define the direction of that rotation axis and q_w defines how far to rotate around it

This is assumed to be a unit quaternion where $\|q\| = 1$

Although four parameters are used to represent quaternion orientation, the time evolution of the quaternion depends only on the three components of angular velocity. We define the angular velocity vector as:

$$\omega = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

The rotational state vector is:

$$Q = \begin{bmatrix} q \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

Its time derivative is:

$$\dot{Q} = \begin{bmatrix} \dot{q} \\ \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix}$$

To determine $\dot{q}$ we express angular velocity as a pure quaternion:

$$\omega_q = \begin{bmatrix} 0 \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

We then use quaternion kinematics to get:

$$\dot{q} = \frac{1}{2} q \otimes \omega_q$$

Which is analogous to $\dot{x} = v$, where instead orientation evolves under angular velocity.

This assumes that angular velocity is expressed in the body frame. Due to the non-commutative nature of quaternion multiplication, the ordering of terms depends on this choice of reference frame. While this choice is arbitrary for the SMD system, we choose a body-frame representation, since the propellers of a drone and resulting forces are defined relative to its body.

Just as force drives translational acceleration in the translation system, torque drives angular acceleration in the rotational system. The rotational equivalent of the applied force u is therefore the applied torque τ .

To complete the rotational model, we must also describe the evolution of angular velocity. This is governed by rotational dynamics:

$$\tau = I_{xx}\alpha$$

$$\frac{\tau_x}{I_{xx}} = \dot{\omega}_x$$

$$\dot{\omega} = \begin{bmatrix} \frac{\tau_x}{I_{xx}} \\ \frac{\tau_y}{I_{yy}} \\ \frac{\tau_z}{I_{zz}} \end{bmatrix}$$

Combining these expressions, we represent the complete rotational dynamics as:

$$\dot{\mathbf{Q}} = \begin{bmatrix} \frac{1}{2}\mathbf{q} \otimes \boldsymbol{\omega}_q \\ \frac{\tau_x}{I_{xx}} \\ \frac{\tau_y}{I_{yy}} \\ \frac{\tau_z}{I_{zz}} \end{bmatrix} \quad (7)$$

Angular velocity is then applied to the quaternion to evolve orientation overtime, directly analogous to how force drives velocity in the translation system.

Unlike the translational system in Eq. 6, the rotational dynamics are inherently nonlinear due to quaternion multiplication. The system fundamentally cannot be expressed in the linear state-space form $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$.

2.2.3 Combined System Representation

We derive the translational and rotational dynamics separately to isolate their distinct mathematical structures, as a unified formulation is less straightforward. As we will study the system as a whole, the state vector must combine both the translations and rotational component vectors into a single state vector representing the full motion.

The combined state is:

$$Y = \begin{bmatrix} x \\ Q \end{bmatrix}, \quad \dot{Y} = \begin{bmatrix} \dot{x} \\ \dot{Q} \end{bmatrix}$$

Substituting the dynamics from Eq. 6 and Eq. 7:

$$\dot{Y} = \begin{bmatrix} Ax + Bu \\ \dot{Q} \end{bmatrix}$$

Expanding this representation then gives the full system state:

$$Y = \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ q \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}, \quad \dot{Y} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \ddot{x} \\ \ddot{y} \\ \ddot{z} \\ \dot{q} \\ \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix} \quad (8)$$

This complete state vector allows both translational and rotational dynamics to be simulated simultaneously as a single system.

2.3 Three-Dimensional Drone System

With the combined three-dimensional state vector established, we now extend the framework to model a drone. In the SMD system, the equation of motion arose from Hooke's law, and internal restoring forces. By contrast, a drone is an input-driven system, where motion is determined entirely by two forces: gravity and rotor thrust. This means that there are no self-stabilizing forces directly within the system.

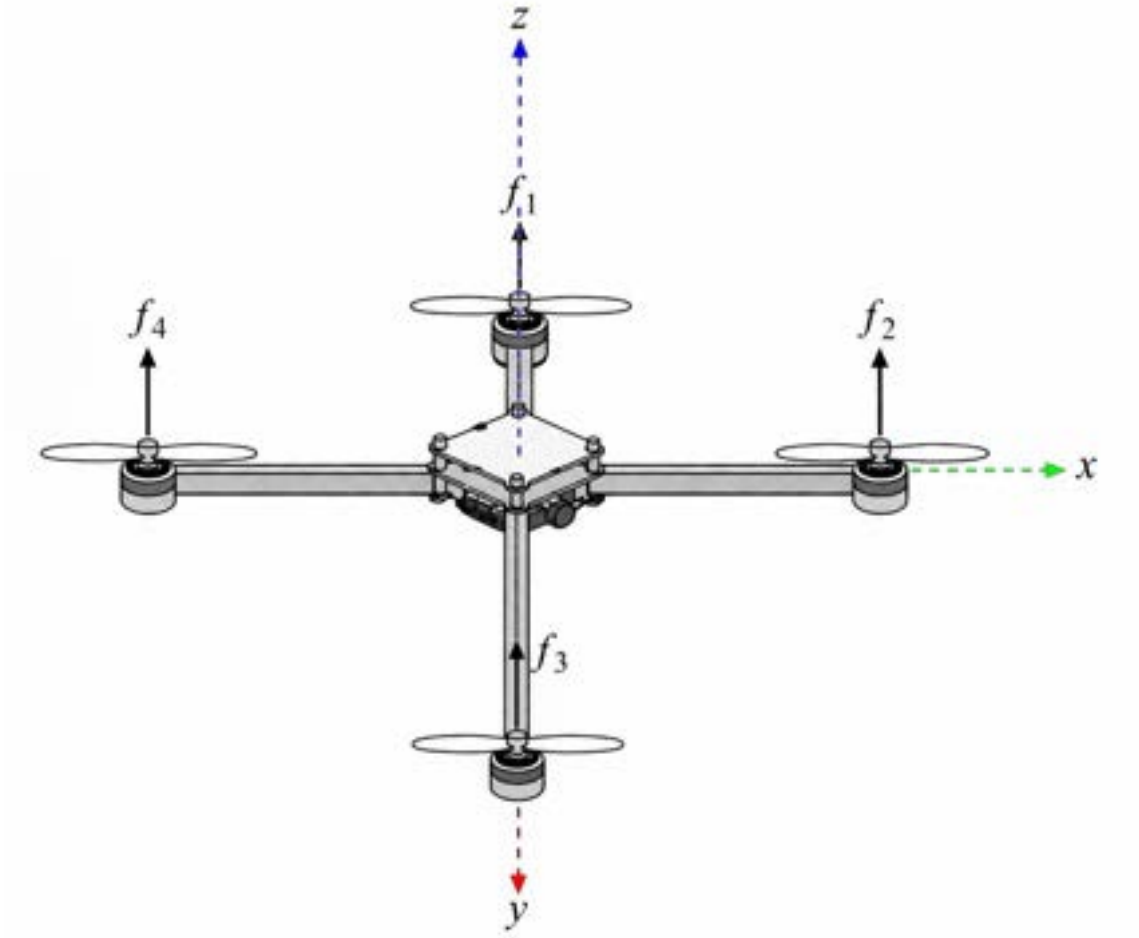


Figure 3

2.3.1 Translational Dynamics

Beginning with translation, we replace the spring-based forces with physically meaningful inputs.

$$m\ddot{\mathbf{x}} = F_{gravity} + F_{thrust}$$

As gravity is a constant downward force:

$$F_{gravity} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix}$$

$F_{gravity}$ is independent of the drone's current rotation, and always accelerates on the $-z$ axis.

F_{thrust} due to the propellers of the drone always acts along the drone's local upward direction. While angular velocity is expressed in the body frame for rotational dynamics,

translational motion is described in the world frame. We therefore rotate the thrust vector using the current orientation to express it in world coordinates.

$$F_{thrust} = q \otimes \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} \otimes q^{-1}$$

The input to the translational system becomes:

$$u = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + q \otimes \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} \otimes q^{-1}$$

Since the system does not have the properties of a spring-damper system, we modify matrix A as well:

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

While matrix B remains unchanged, as it describes how external forces act on the system.

2.3.2 Rotational Dynamics

In the drone, torques are no longer free independent inputs as in the SMD. Instead, they arise directly from differences in rotor thrust. This is where F and T – previously independent – become coupled through the rotor forces.

We model rotation through the forces of each individual propeller. Let each motor produce thrust f_1, f_2, f_3, f_4 , where the net thrust is:

$$T = f_1 + f_2 + f_3 + f_4$$

To rotate the drone, we need to account for the torque from imbalances in rotor forces. Roll and pitch torques are generated by differences in thrust across opposing arms:

$$\tau_x = r(f_2 - f_4), \quad \tau_y = r(f_3 - f_1)$$

r – distance from center of mass to each rotor

Yaw torque results from reaction forces:

$$\tau_z = c(f_1 - f_2 + f_3 - f_4)$$

c – yaw torque coefficient of motors

We then substitute these torque expressions into Eq. 7:

$$\dot{Q} = \begin{bmatrix} \frac{1}{2}q \otimes \omega_q \\ \frac{r(f_2 - f_4)}{I_{xx}} \\ \frac{r(f_3 - f_1)}{I_{yy}} \\ \frac{c(f_1 - f_2 + f_3 - f_4)}{I_{zz}} \end{bmatrix} \quad (9)$$

With this, we define the complete drone dynamics as a coupled translational and rotational system, fully described by first-order differential equations.

2.4 Numerical Integration

For linear systems such as a SMD, analytical solutions exist. However, for the nonlinear rotational dynamics governed by quaternion multiplication, and for drone model where rotation affects translation, no closed form solution is available. We therefore approximate the solution numerically.

The models developed so far can be expressed as systems of first-order differential equations of the form:

$$\dot{x} = f(x, u)$$

Solving this differential equation enables us to understand how the system evolves over time. For this paper, we use Euler's method, which updates the system state using the current derivative according to:

$$x_{k+1} = x_k + \Delta t \cdot f(x_k, u_k)$$

The choice of time step Δt determines the accuracy of the approximation, with smaller values providing improved fidelity at the cost of increased computational effort.

For linear systems of the form:

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

The function can also be expressed as:

$$x_{k+1} = x_k + \Delta t \cdot (Ax_k + Bu_k)$$

We will use Euler's method in Section 5 to simulate and compare the behavior of the systems under various conditions.

3 The Main Result

3.1 Unified Drone System

We now present the unified dynamical model of the drone, combining the translational and rotational components derived in section 2.3. It is expressed as a set of first-order differential equations governing the evolution of the full state of the drone.

The complete system state of the drone is:

$$Y = \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ q \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

The system dynamics are described by:

$$\dot{Y} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \ddot{x} \\ \ddot{y} \\ \ddot{z} \\ \dot{q} \\ \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix}$$

This can be written compactly as:

$$\dot{Y} = \begin{bmatrix} \dot{x} \\ a_{\text{trans}}(Y, u) \\ a_{\text{rot}}(Y, u) \end{bmatrix} \quad (10)$$

Where the translational state derivative is given by:

$$\dot{x} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix}$$

The translational dynamics is given by:

$$a_{\text{trans}} = \frac{1}{m} \left(\begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + q \otimes \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} \otimes q^{-1} \right)$$

And the rotational dynamics is given by:

$$a_{\text{rot}} = \begin{bmatrix} \frac{1}{2} q \otimes \omega_q \\ \frac{r(f_2 - f_4)}{I_{xx}} \\ \frac{r(f_3 - f_1)}{I_{yy}} \\ \frac{c(f_1 - f_2 + f_3 - f_4)}{I_{zz}} \end{bmatrix}$$

This defines the complete drone dynamics as a coupled, nonlinear system of first-order differential equations, representing a direct extension of the modeling framework developed from simpler systems, and which can be numerically propagated using the integration approach described in section 2.4.

4 Discussion

The progression from the one-dimensional SMD model to the drone model shows that simpler systems can provide a useful framework for understanding more complex dynamics. The SMD system establishes the basic modeling pattern used throughout the paper: define a state vector, express the governing equation as a first-order system, and then propagate the state to simulate it. This structure remains useful even after the system becomes higher-dimensional and nonlinear. As a result, the simpler model serves as the foundation for interpreting the final drone model.

The three-dimensional extension also shows that different parts of the model increase in complexity at different rates. Translational motion can be extended from the one-dimensional case by applying the same dynamics along each spatial axis. Rotational motion, however, requires a different representation because orientation evolves through angular velocity and depends on the chosen reference frame. We use quaternions, which avoid the discontinuities and gimbal-lock problems associated with Euler angles, but they also introduce nonlinear behavior through quaternion multiplication. This changes the framework to a coupled nonlinear system.

The drone model highlights the main behavioral difference between passive and input-driven systems. In the SMD model, the spring and damper naturally produce restoring behavior, so the damped system tends toward equilibrium without requiring an external controller. The drone does not have this same passive stabilizing structure. Its motion is determined by thrust, torque, and gravity, so stable behavior must be produced through feedback control. The same state-space framework can describe both systems, but stability must be interpreted differently if the system depends primarily on control inputs.

5 Simulation Results

Simulation Results for 1D SMD

To evaluate the behavior of the system, a numerical simulation of the system from section 2.1 is performed using Euler's method from section 2.4. The resulting time evolution of position, velocity, and acceleration for various initial conditions is shown in Figs. 4–7, with the corresponding parameters provided in each figure.

Due to the non-zero damping coefficients, all systems are stable and always converge onto the equilibrium value given enough time. This establishes the fundamental dynamics that is later extrapolated to the higher-dimensional models.

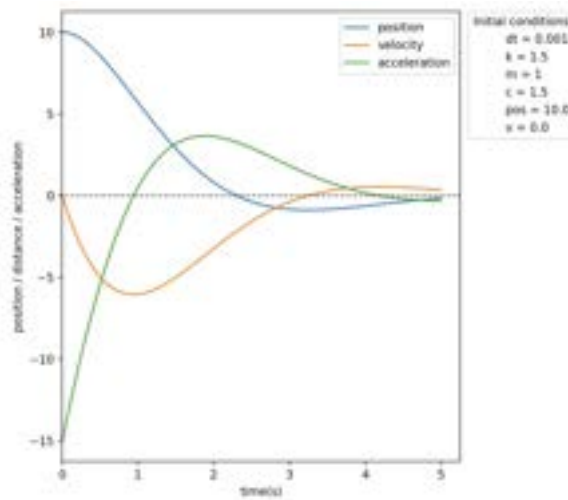


Figure 4

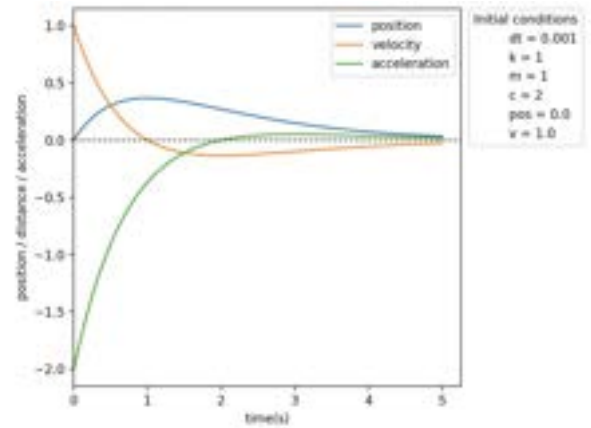


Figure 5

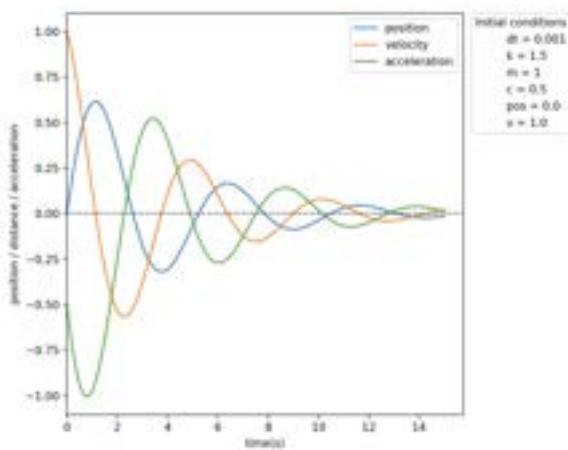


Figure 6

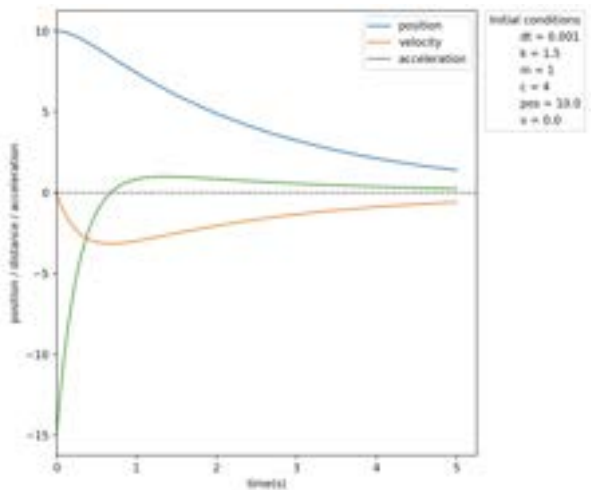


Figure 7



Simulation results for 3D SMD

Next, the three-dimensional SMD model from section 2.2 was implemented using the same numerical framework from section 2.4. The resulting time evolution of position and velocity in three dimensions is shown in Fig. 8.

Each spatial axis follows the same translational dynamics as the one-dimensional system, resulting in a three-dimensional trajectory formed by the combination of otherwise independent motions. This illustrates how the higher-dimensional system is a direct extension of the 1D model. The rotational dynamics influence the orientation of the system and the direction of applied forces, result in a continuously increasing angular velocity due to a lack of damping.

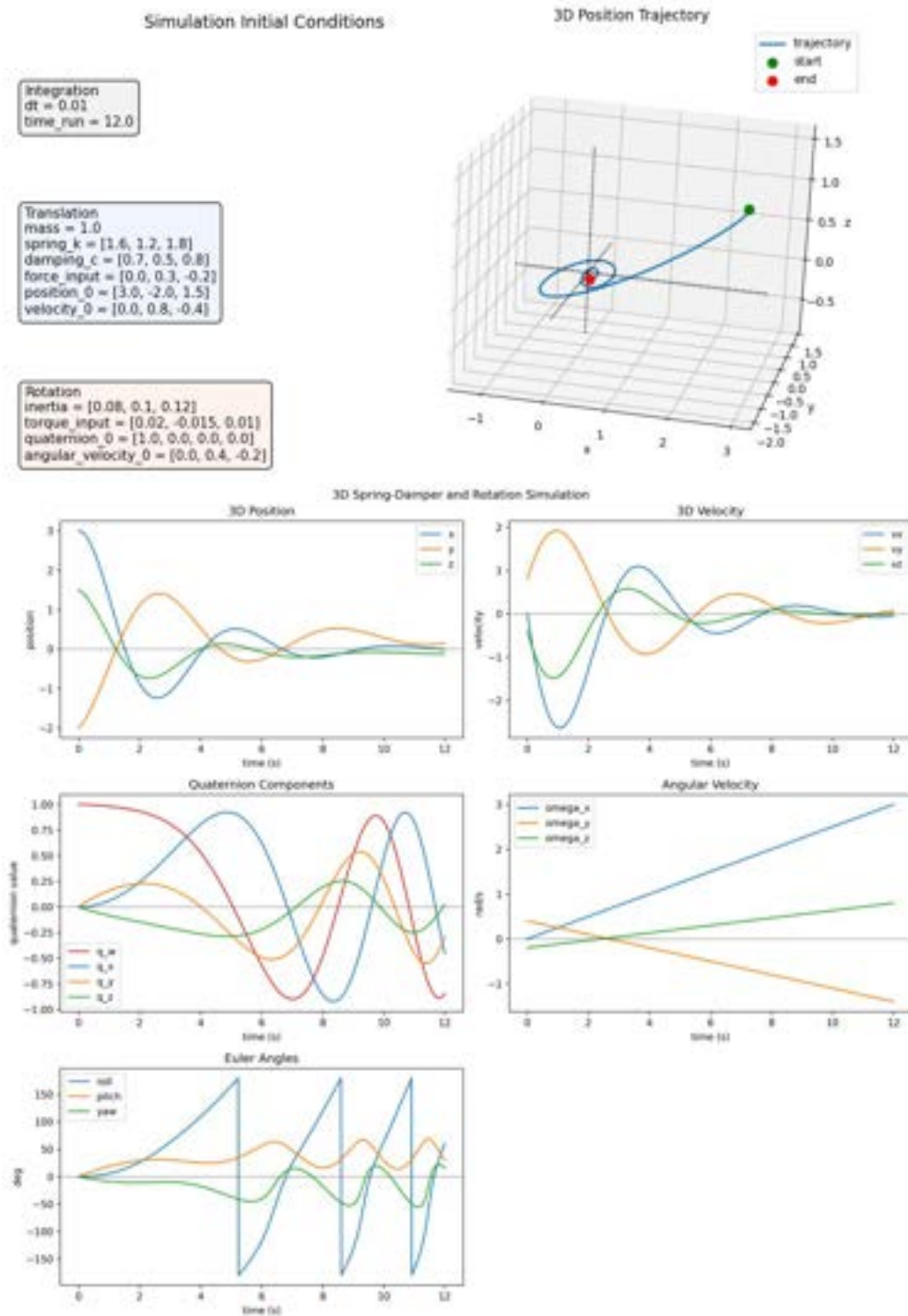


Figure 8

Simulation Results for 3D Drone System

To evaluate the behavior of the drone model, the unified state-space system from 3.1 was simulated. The resulting time evolution of position, velocity, orientation, and rotor forces is shown in Fig. 9.

Unlike the SMD system, the drone is a fully input-driven system with no passive restoring forces. Stability is therefore not an inherent property of the system but must be achieved through control.

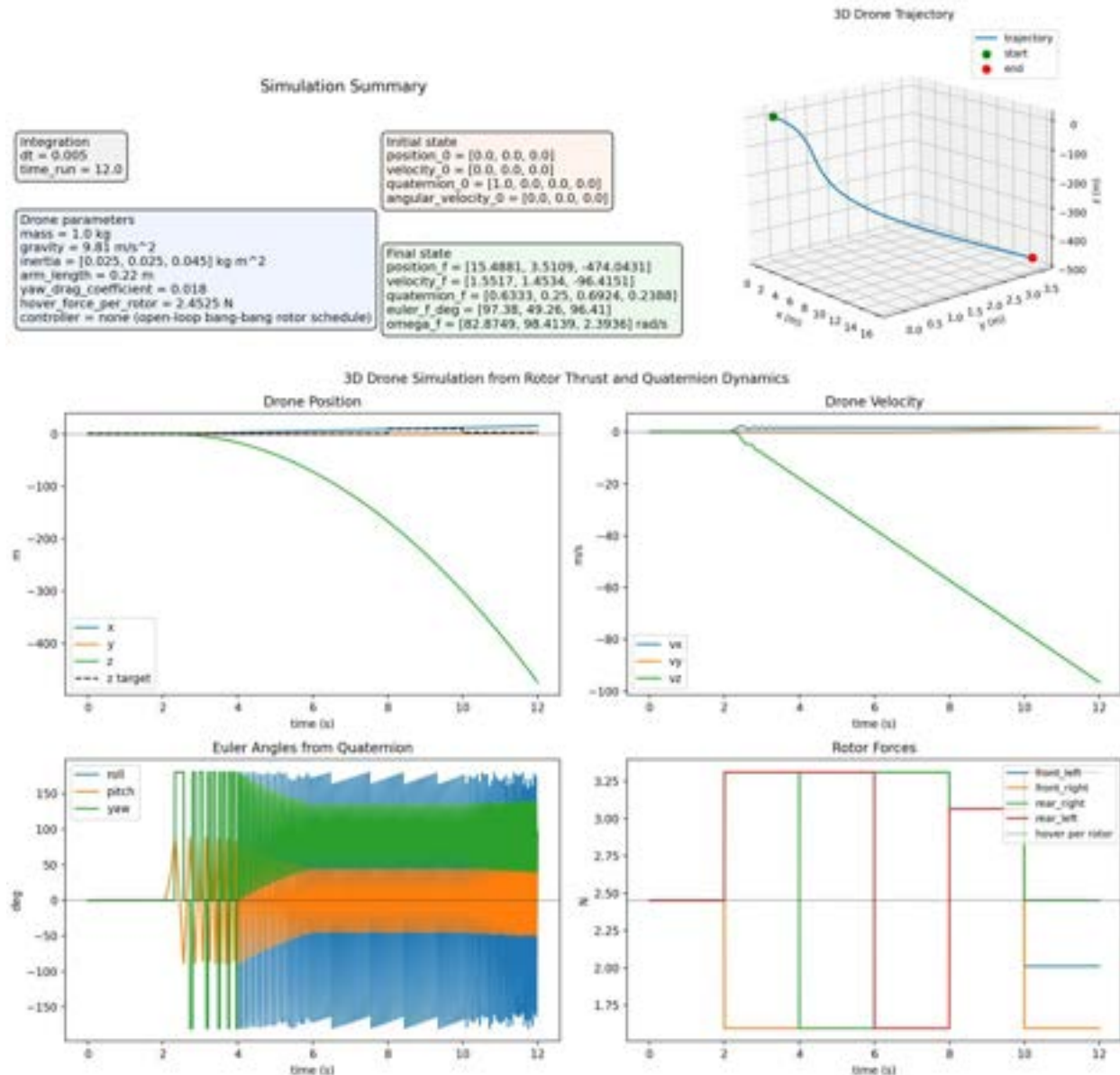
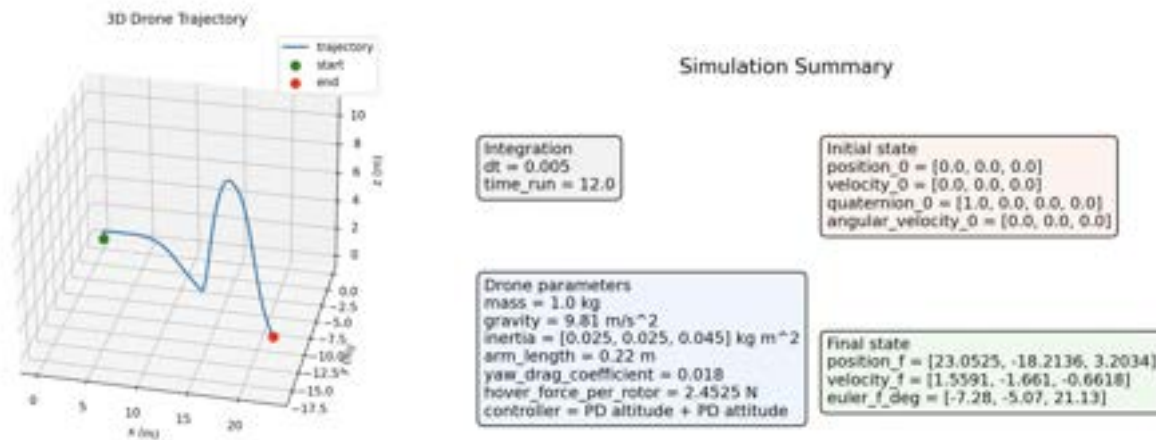


Figure 9

As expected, the open-loop system exhibits unstable behavior, with the state diverging over time. In contrast to the SMD models, the drone contains no inherent restoring factor, and its behavior is entirely determined by the applied inputs.

Despite this difference, the underlying state-space form remains consistent with the earlier models. We have extended the framework from a 3D SMD model to a more complex system. When a simple feedback controller is introduced, the system remains bounded and exhibits stable behavior. While the control is not perfect, it prevents rapid divergence and allows for meaningful analysis. A basic PD controller is implemented in Fig. 10 for the same system.



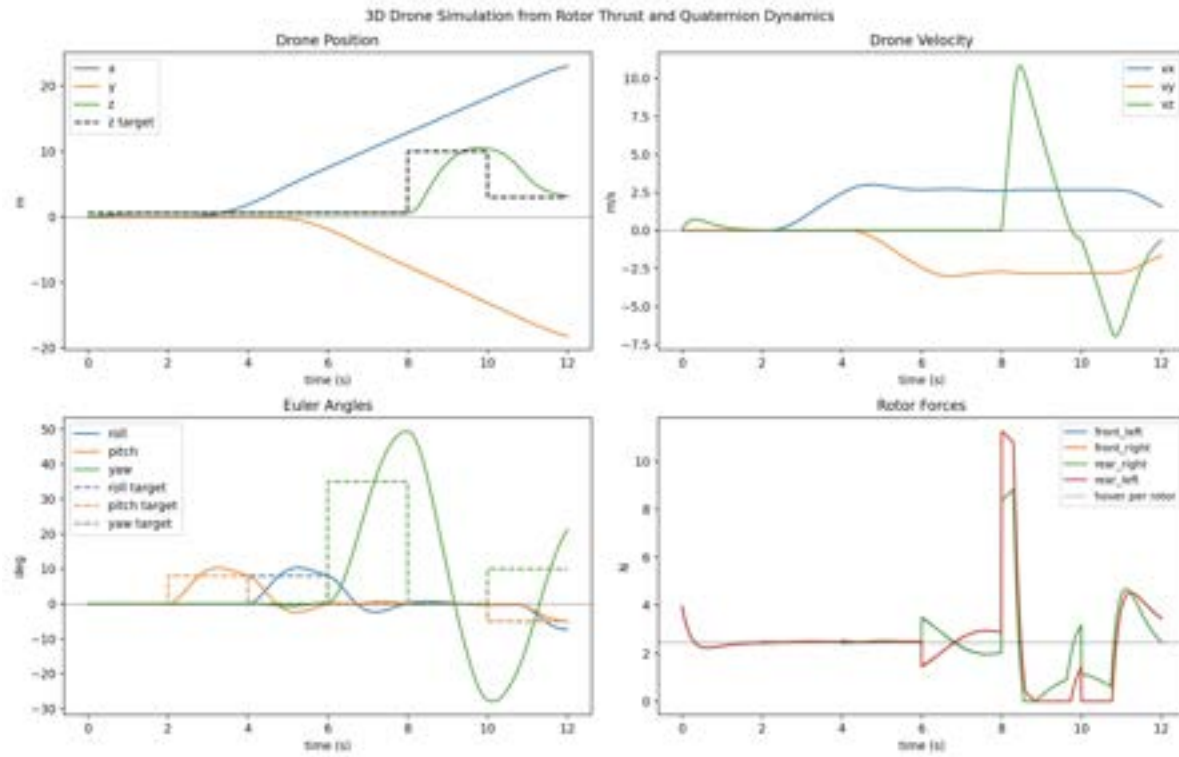


Figure 10

6 Conclusions and Future Work

We developed a progressive modeling framework beginning with a one-dimensional spring-mass-damper system and extending it to a three-dimensional drone model. The 1D SMD model helped us introduce a state-space-representation of a second-order differential equation, establishing the basic structure of the system. The 3D SMD model then extended that model across 3 spatial axes and introduced quaternion kinematics to describe rotational motion. Finally, these dynamics were modified to fit a drone model as an input-driven system controlled by thrust, torque, and gravity. Each model was simulated using Euler's method, showing the contrast between naturally stable SMD behavior and the control-dependent behavior of the drone. The main result is the demonstration that a complex nonlinear model can be approached through a sequence of simpler dynamical systems.

This modeling progression can be extended in two main directions. First, the approach can be applied to other complex systems, using simple linear models before gradually introducing higher-dimensional motion, nonlinear dynamics, and control inputs. The framework can serve as a teaching tool for moving from linear state-space systems to more complex nonlinear control systems. Second, the drone model itself could be made more realistic while preserving the same progressive structure. Effects such as aerodynamic drag, motor limits, actuator saturation, wind disturbances, sensor noise, measurement uncertainty, and motor response delay could be introduced in addition, without losing the interpretability provided by the simpler models. This would preserve the central goal of the work: to make complex dynamics easier to understand by building them from simpler, more familiar models.



References

- [1] N. S. Nise, Control Systems Engineering, 7th ed. Wiley, 2014.
- [2] J. Solà, “Quaternion kinematics for the error-state Kalman filter,” arXiv preprint arXiv:1711.02508, 2017.
- [3] R. Kumar, M. Bhargavapuri, A. M. Deshpande, S. Sridhar, K. Cohen, and M. Kumar, “Quaternion feedback based autonomous control of a quadcopter UAV with thrust vectoring rotors,” in Proc. 2020 American Control Conference (ACC), Denver, CO, USA, 2020, pp. 3828–3833.