UAV Dynamics and Electric Power System Modeling and Visualization using Modelica and FMI

Meaghan Podlaski
Graduate Research Assistant
Rensselaer Polytechnic Institute
Troy, NY, USA

Luigi Vanfretti
Associate Professor
Rensselaer Polytechnic Institute
Troy, NY, USA

Hamed Nademi
Research Scientist
Rensselaer Polytechnic Institute
Troy, NY, USA

Hao Chang
Student
Rensselaer Polytechnic Institute
Troy, NY, USA

ABSTRACT

This paper presents an object-oriented, equation-based framework for multi-engineering modeling of a quadrotor UAV, which includes the rigid body dynamics, simplified aerodynamics, gyroscopic effects, electrical power system and battery losses, and DC motor dynamics. An open-source drone modeling library is introduced by explaining the mathematical models and multi-domain components used to model the drone. Animation and visualization techniques for the drone using CAD models are also introduced and explained. The proposed drone model is simulated under different flight scenarios using motor and power system models with different levels of detail, aiming to provide better means for design and understanding, of multi-engineering aspects of UAVs. This model provides a foundation for future UAV open-source model development, electrified power propulsion design, visualization and interaction, and system identification.

NOTATION

b	Propeller friction force
f	Linear force
K_{bat}	Polarization Resistance
K_{τ}	Motor current to torque constant
K_f	Propeller force constant
I_a	Armature current
I_e	Excitation current
J_p	Propeller inertia
J_r	Rotor inertia
L_e	Excitation inductance
MAV	Multirotor aerial vehicle
MSL	Modelica Standard Library
n	Turns ratio
np	Number of battery cells in parallel
ns	Number of battery cells in series
P_p	Propulsive power
Q	Current battery charge content
Q_{bat}	Total battery charge capacity
Q_{cell}	Cell charge capacity
SoC	State of charge
vCell	Battery cell voltage
$vCell_{max}$	Maximum battery cell voltage
τ	Torque
τ_h	Aerodynamic thrust
τ_o	Aerodynamic torque
τ_e	Electrical torque
UAV	Unmanned aerial vehicle
V_a	Armature voltage
ω	Angular speed
ψ_e	Excitation flux

Presented at the VFS International 76th Annual Forum & Technology Display, Virginia Beach, Virginia, October 4–6, 2020.

INTRODUCTION

Motivation

Increase in demand for high-speed mobility, sustainability, and profitability requires advancements in aviation technologies. This includes the research and exploration of intelligent autonomous flying machines, specifically multi-domain Unmanned Aerial Vehicle (UAV) drone modeling. Simulation-based studies are extremely valuable to determine which concepts and methods meet requirements and specifications. Creating physical prototypes for complex multi-engineering systems can also be costly and difficult; while opportunities for testing using a physical system are limited. As a result, well-defined, reliable models are essential for the development of new aircraft systems and aircraft technologies.

This multi-domain model-based systems engineering method is implemented herein for a drone, which has been created using the object oriented modeling language, Modelica. Multi-domain models were created to represent each aspect of the drone, specifically focusing on the mechanical, electrical, and control domains. This paper will introduce the Modelica library created for the drone modeling. It shows these models at varying levels of detail under different operating conditions, and discusses the importance of multi-domain models for the design of the electrified propulsion power system.

Related Works

The growth of interest in this area has led to the modeling of UAV primarily in MATLAB such as (Ref. 1), where the model only covers a quadrotor case without option for model visualization. Other previous works in this area include Kuric (Ref. 2) to model a quadcopter, including the dynamics of the motor using Modelica. It mainly focuses on multirotor aerial vehicle (MAV) dynamics modeling, while assuming ideal power consumption and operation. All of the dynamics are reduced to one domain to a linear, mathematical model. In this paper, the drone model is expanded to consider a multi-domain model with a non-ideal power system and switching power electronic components.

(Ref. 3) introduces a method for drone PID controller modeling using Modelica. The drone model presented in (Ref. 3) assumes that the body is rigid and symmetrical, the center of mass and body fixed frame origin, and the force of each propeller of the aircraft is proportional to the square of the speed of the propeller. The model presented in this paper does not make these ideal assumptions and accommodates for the non-ideal behavior and different airframe structures.

Quadcopter modeling and simulation has been investigated and analyzed using Matlab and Simulink as per Luukkonen (Ref. 4). Both of these studies primarily focus on the analysis of the dynamical model of the quadrotor. The analysis in (Ref. 5) provides the most complete work regarding micro air vehicles (MAVs), where the dynamics, advanced state estimation, control and motion planning algorithms for the MAV are derived. This model strictly focuses on the linear dynamic mathematical models, while the models presented in this paper focus on more detailed dynamic models of the drone's power system, including motor models considering electrical effects and the effects of a non-ideal power source on the system.

Paper Contributions

The main contributions of this paper are:

- Propose an open-source library consisting of multi-domain components used to model a drone.
- To show the details and mathematical models for each component used as well as results of simulation and animation of these models.
- To perform studies highlighting the importance and impact of adequately modeling the power system of the drone.
- To illustrate the importance of multi-domain modeling for closed-loop system performance.

Paper Organization

This paper utilizes the Modelica modeling language to develop the multi-domain model of a drone and all of its respective control components. The paper is organized as follows. First, the models and components used to create the drone are discussed in detail, showing the mathematical modeling for each component in the drone. Next the animation and simulation of the drone model is discussed. The model is then studied for reference tracking for an ideal flight path focusing on the time specification responses and energy metrics. The impact of the power system modeling level of detail on closed loop dynamics is discussed, as well as models including a payload and the effect of a varying payload on the model.

MODELICA DRONE LIBRARY

The drone library and models were implemented in Dymola using the Modelica language. The library is open-source and can be found at (Ref. 6). It is divided into different packages, or subdirectories, consisting of blocks, sensors, electrical components, mechanical components, and examples as shown in Figure 1. Each of these packages contain the components needed to build the drone model. The library also uses components from the Modelica Synchronous Library (Ref. 7), Modelon Base Library (Ref. 8), and Modelica Standard Library (Ref. 9) to run certain model variants. The Modelica Standard Library (MSL) is a library consisting of standardized models maintained by the Modelica Association (Ref. 9). The drone model is based on the Otus Quadcopter (Ref. 10). The model's main limitations arise from assuming that the drag and power of the quadrotor is not affected by the altitude of flight.

Blocks

The **Blocks** package contains sources, control models, math functions, and routing functions. The **Sources** are the customized signal blocks that can be used to create flight paths for the drone. This includes a circular and straight line flight path. In addition to using these pre-existing flight paths to control the drone, data obtained from a physical system experiments can be defined in a table consisting of time and physical position. The **Control** package contains models for a discrete PID controller used in the drone controller. These models utilize the Modelica Synchronous Library (Ref. 7) to precisely define and synchronize the sampled data components with different sampling speeds in the controller and to improve simulation speed. The **Math** package has the customized math functions used in the controller, and the **Routing** package features functions for expanding the data from sensors to be used in the three dimensional plane.

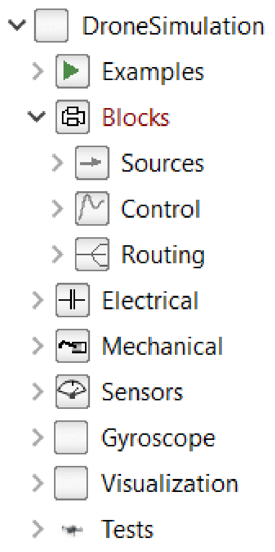


Figure 1. Library package setup.

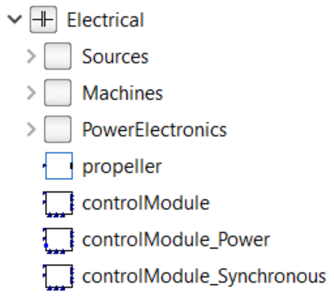


Figure 2. Electrical package setup.

Sensors

Custom sensor functions for the drone that track the position and acceleration of the drone are in the **Sensors** package. These sensors are shown in Figure 5 as the `GPS` and `Accelerometer` components.

The `GPS` model uses a relative position sensor to tracks the position of the drone with respect to the ground. It uses that information to change the flight path.The `Accelerometer` uses a relative angle sensor to track the pitch, yaw, and roll of the drone's airframe.

Electrical

The **Electrical** package contains models for the DC machines and power system used in the drone model, split into subpackages as shown in Figure 2. The **Sources** subpackage contains models for the batteries used in the non-ideal power system, **Machines** subpackage has models for the motors driving the propellers, and **Power Electronics** includes converter and switch models for the drone power system.

Sources: The **Sources** subpackage has the models for the drone power system. The power system is modeled after the Otus Quadcopter (Ref. 10) power system, as it is the drone currently available for experiments in the author's lab. A sample of the FFT Gyroscope is shown in Figure 3. Figure 4 shows the connections between the battery and sensors for the power system of the drone. The battery is connected to a DC/DC step down converter to provide 5V connections to power the Raspberry Pi, Pixhawk flight controller, and LCD Display. The Otus tracker is not included in the drone model because the experimental test bed is a gyroscope and the XYZ coordinate position of the drone do not change. The motors operate at 11.1V from the battery.

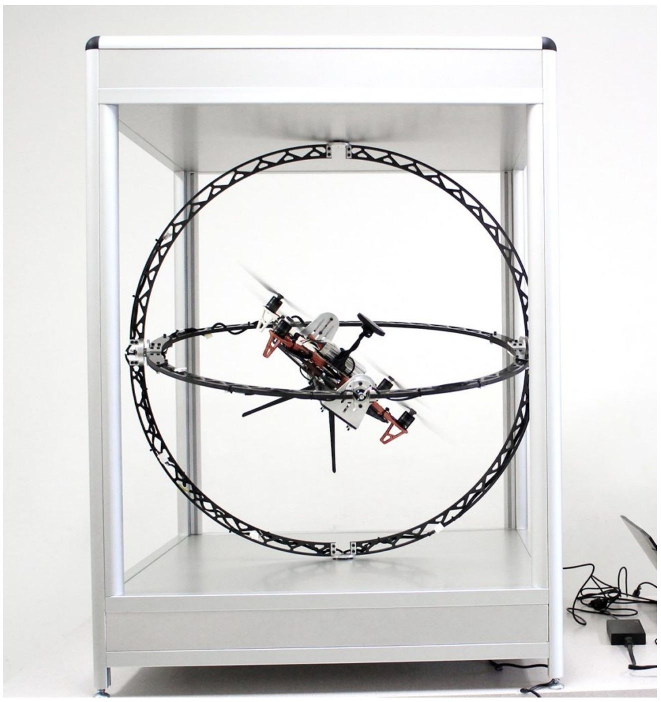


Figure 3. FFT gyroscope test bed setup (Ref. 11).

The power system in Figure 4 is divided into the battery system in Figure 5, which consists of the DC/DC step-down converter and battery. The outputs of the DCDC step down converter provides 5V to the Raspberry Pi, LCD display, and Pixhawk. The output connectors of the battery provide 11V to the motors. The rest of the components in Figure 4, such as the motor and Raspberry Pi, are modeled in the rest of the drone system shown in Figure 5.

The battery model for the power system is shown in Figure 6, which is from the Modelon Base Library. It contains an internal resistance and stack voltage that discharges according to Equations 1, 2, 3, and 4, which are the extended Shepherd equations for battery pack EMF (Ref. 12). Equation 5 determines the charge capacity of the battery if there are cells placed in parallel.

$$vCell = vCell_{max} - \frac{K_{bat}}{SoC} \tag{1}$$

$$\frac{dQ}{dt} = i \tag{2}$$

$$SoC = \frac{\min(Q, Q_{bat})}{Q_{bat}} \tag{3}$$

$$v = vCell * ns \tag{4}$$

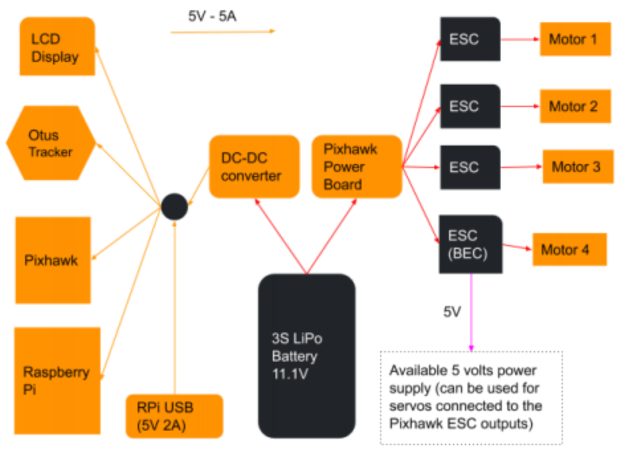


Figure 4. Otus Quadcopter power system (Ref. 10).

$$Q_{bat} = Q_{cell} * np \tag{5}$$

The battery is a Li-Ion 18650 with three cells in series, as that is the same battery required to power the Otus Quadcopter. This is modeled such that the maximum, nominal, and minimum voltages of the cell are 4.2 V, 3.7, and 3.0 V respectively. In Equation 3, Q_{cell} is 9360 C for a Li-Ion 18650 battery. The internal resistance of the battery shown in Figure 6 is 0.1 Ω.

Machines: Multiple motor models of varying degrees of complexity are included in the library. These models include different types of DC machines along with a simple, ideal motor.

The simple motor only utilizes torque, linear force, motor speed, and current to control the motor according to Equations 6, 7, and 8. There are no electrical, thermal, or mechanical losses considered in this model.

$$\tau = K_\tau * i \tag{6}$$

$$J_p \frac{d\omega}{dt} = \tau - b * \omega \tag{7}$$

$$f = K_f * \omega \tag{8}$$

Most UAV systems use brushless DC motors for their high efficiency and high power to size ratio, so the most complex DC motor used in the model is a permanent magnet DC machine. The model for the permanent magnet DC machine is shown in Figure 7, which models the electrical, rotational, and thermal behaviors in the machine. The dark blue lines represent electrical connections in the machine. The gray lines represent rotationally linked components, which covers behavior such as frictional losses due to the air gap in the machine and the rotation of the machine. These domains are connected through the air gap of the motor creating a magnetic field to turn the rotor.

The air gap for this machine has an electrical flux that is linearly dependent on the excitation current, shown in Equation 9. The armature voltage, which is the voltage from the `Speed`

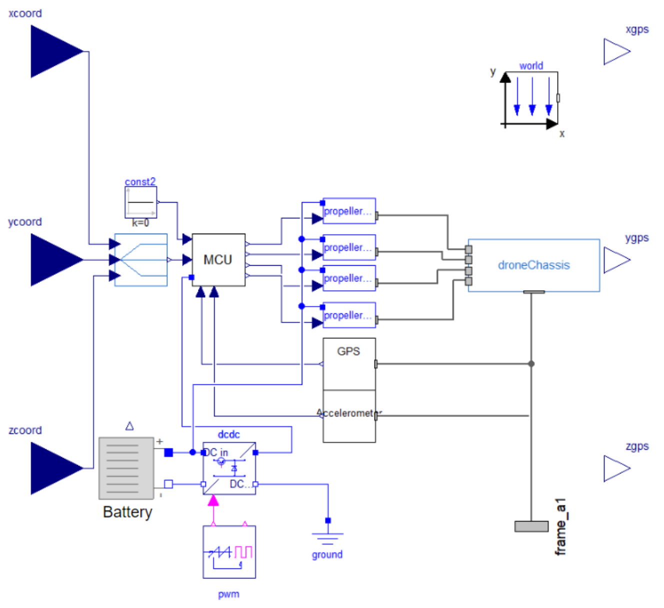


Figure 5. Complete drone model consisting of propellers, motor, controller, and chassis with battery power system. Inputs come from x, y, and z coordinate location.

`Controller` component in Figure 16, relates to the speed of the machine using Equation 10. Equation 11 determines the torque applied to the propellers.

$$\psi_e = L_e * I_e \tag{9}$$

$$V_a = n * \Psi_e * \omega \tag{10}$$

$$\tau_e = -\tau = n * \Psi_e * I_a \tag{11}$$

The mechanical model in Figure 7 includes the inertia of the stator (which is fixed to the body) and the inertia of the rotor, J_r. This is important, as it allows to specify both stationary and rotational masses that will ultimately impact the drone's closed-loop behavior.

Power Electronics: The **Power Electronics** sub package contains the models for the DC/DC converters and switches used in the drone. This includes the buck DC/DC converter used in the power system, which is shown in Figure 8. Averaged converter model components are also available in this package, but they are currently not used in the studies in this paper.

Controllers

The controller for the drone is shown in Figure 9. It consists of multiple discrete PID controllers, which are configured using the Modelica Synchronous library to ensure fast simulation compiling and integration times for models with discrete control components that are being sampled at different rates. There is also a controller included that does not utilize the Modelica Synchronous library if it is unavailable or more conventional control modeling is desired. The system

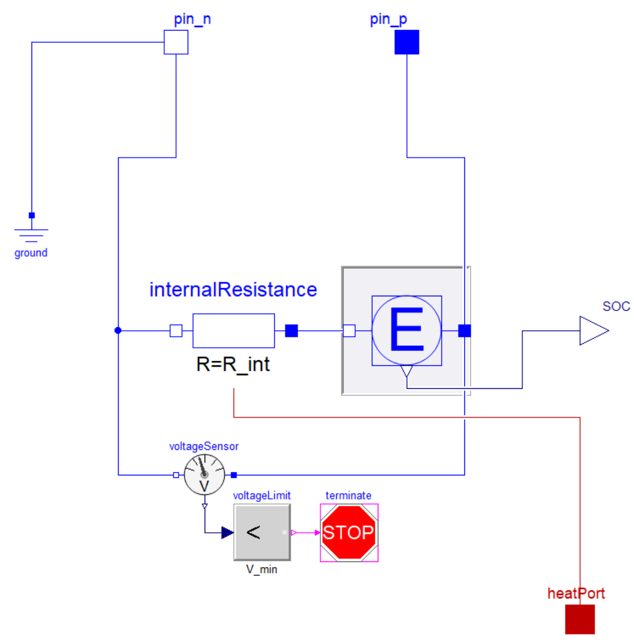


Figure 6. Battery from Modelon Library Suite (Ref. 8).

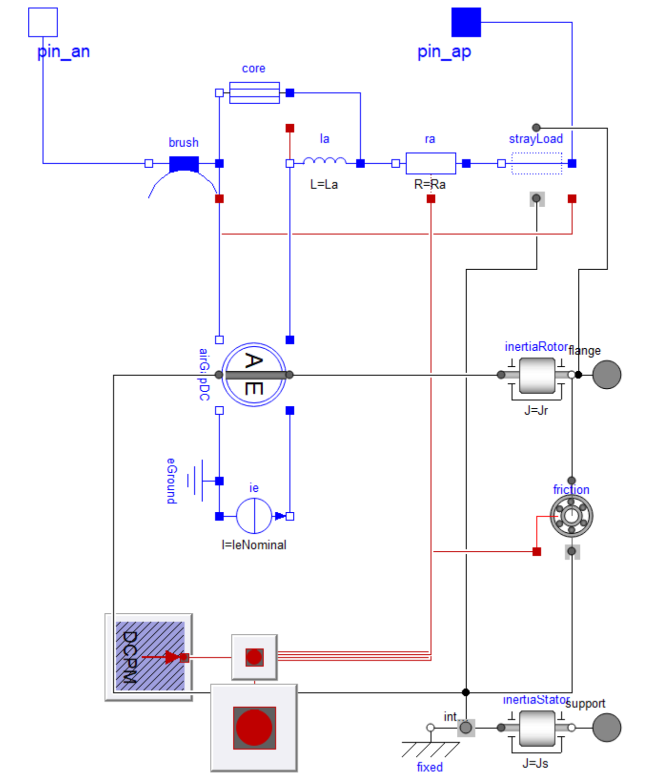


Figure 7. MSL permanent magnet DC machine model (Ref. 9).

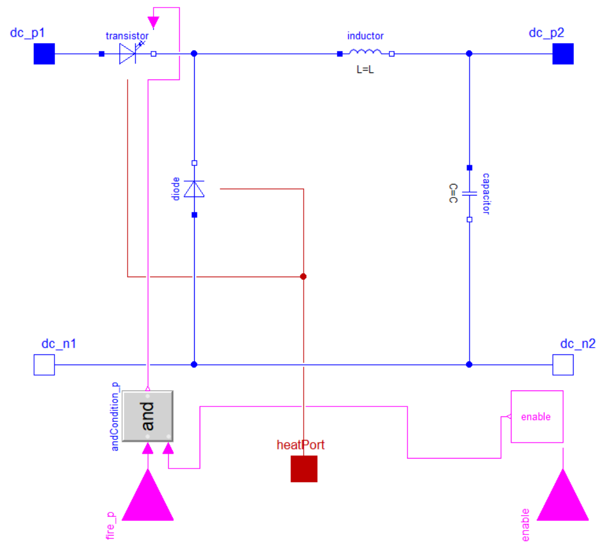


Figure 8. Buck converter model in Dymola.

forms a closed loop with tracking of the coordinate position, pitch yaw, and roll of the drone.

In Figure 9, the `GPS[]` input translates the XYZ coordinate position of the drone's center of mass into three vectors that provide negative feedback to the position vectors from the user input `position[]`. This ensures that the coordinate position of the drone can follow the user input with a small error. This signal is then used as the input for another PID controller with a reference signal of the relative angles in the X, Y, and Z direction of the drone. This determines the pitch, roll, and lift of the drone. The `yaw` input of the controller uses a reference yaw signal compared to the gyroscope measurement. In the cases presented in this paper, the yaw is desired to be 0 to prevent the drone from spinning around the Z axis. Each output of the controller, `y`, `y1`, `y2`, and `y3` takes into account the lift and the yaw of the drone; propellers diagonal to each other account for either pitch or roll for balanced operation.

When the power system is accounted for in the model, the power consumption of the controller is modeled as a resistive loss. This is shown as the electrical connection pin, `pin`, using the output of the DCDC converter.

The speed controller is also included in the **Controllers** package. It is used in the motor model to adjust the position from the controller by the voltage supplied by the battery. In Figure 16, this component is the `Speed Controller` connected to the electrical inputs of the DC permanent magnet machine with external connections to `position` from the controller and `p1` from the battery. The contents of the speed controller is shown in Figure 10. The `const` block is a reference voltage representing the nominal voltage of the battery. As the battery discharges, the potential sensor adjusts the ratio of position. This function assumes that the battery voltage and motor power reduction are linear.

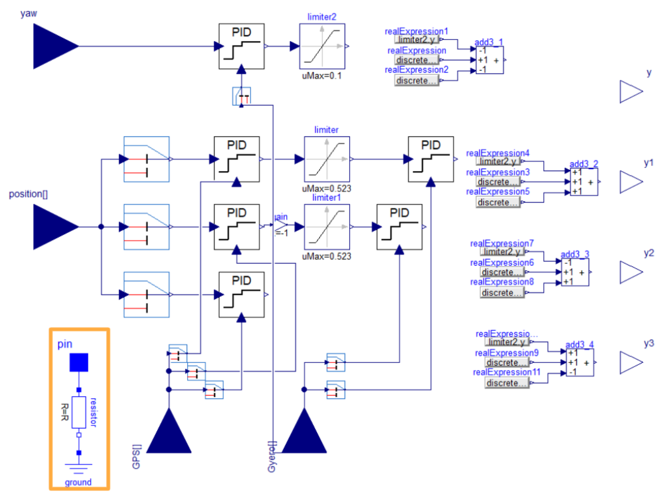


Figure 9. Drone controller model in Dymola. The electrical pin, `pin`, represents the electrical losses in the model. The electrical loss model of the controller is denoted by the orange box.

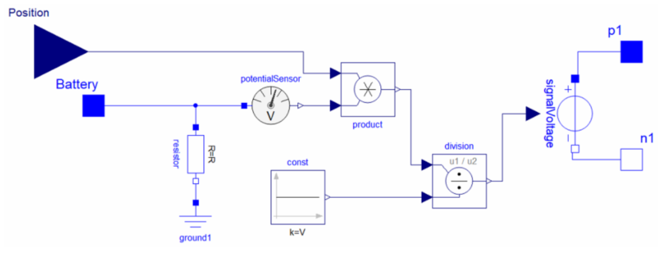


Figure 10. Speed controller model in Dymola. The input `position` is the signal from the controller to determine the torque produced by the motor, the input `battery` is an electrical input containing the voltage and current output from the battery.

Mechanical

The **Mechanical** package contains the models for the quadcopter propeller, chassis, rotor, and motor. The sub-packages in the **Mechanical** package is shown in Figure 12. The components included in this subpackage also are used for the animation and visualization of the drone.

Blades: The **Blades** package contains the models of the propeller blades. The propeller blades are modeled as two multi-body masses coupled mechanically to the rotor. These components are from the MSL's Multi-Body Library (Ref. 13). Figure 11 shows the model used to simulate and animate the blades. In this model, the relationship between angular speed of the propeller shaft and the thrust is assumed to be linear.

In Figure 11, the `fixedShape` component links to a 3D object (.STL) file, which will be used to animate the drone's flight. This 3D object file can be changed to represent the blade configuration of any type of drone. The `bodyShape` are single-point mass components that couple to the rest of the system as a function of 3D orientation, cut force, and cut torque.

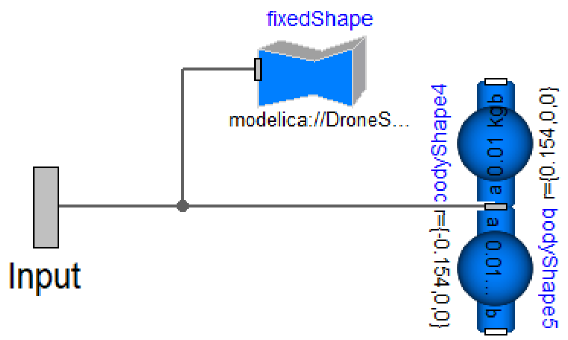


Figure 11. Blade model in Dymola.

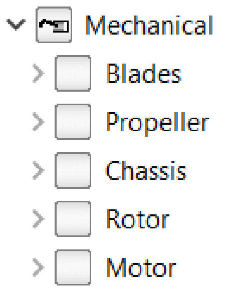


Figure 12. Mechanical package setup.

Propeller: The **Propeller** models contain the motor and the propeller blades controlled by the relative angular velocity of the blades. The propeller model in Dymola is shown in Figure 15. In Figure 15, the propeller system is shown to be mechanically linked between the motor, rotor, and propeller, while using a real signal that provides the reference signals used to control the motor. This model can be used to rotate the propeller both clockwise and counterclockwise by adjusting the value of `gain1`. The drone chassis is modeled as a point mass connected to the airframe, which will have the propellers connected at the end. These models are reusable as the weight and size of the chassis can be easily altered to fit the parameters of any quadcopter. Figure 15(A) is used when an ideal power source is modeled, while Figure 15(B) is used when a non-ideal power system model is included in the system.

Another notable feature seen in the drone model is that multi-domain interfaces can be used. In Figure 16, for example, the blue connections represent real signals and the gray connections represent multi-body mechanical variables. This allows for better organization of the models and for different variables to be interfaced between components. For example, in the electrical domain, these interfaces to couple the voltage and current between connected components. When a non-ideal voltage source is used, the propeller will be represented by Figure 16 and `p1` must be connected to an external power source.

Chassis: The chassis sub-package contains the model of the drone airframe. Similar to the blade models, the chassis is represented by single point masses with a `fixedShape` component to integrate the 3D object file of the drone chassis into the

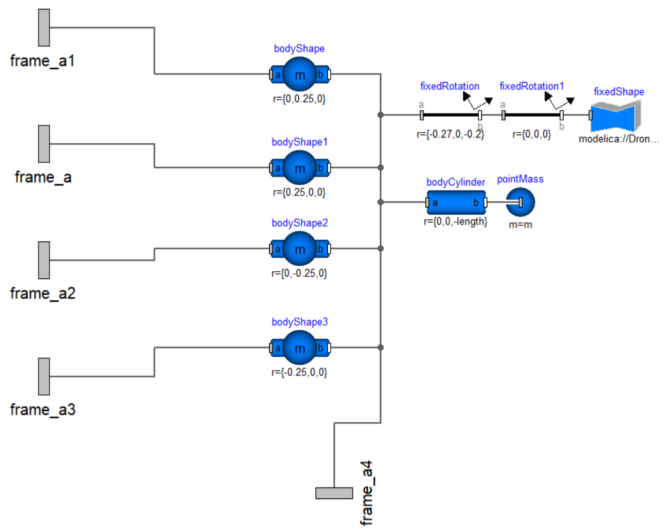


Figure 13. Chassis model in Dymola.

rest of the model. The chassis is mechanically linked to the propellers, as shown in Figure 17, where each of the `frame` components are connected to a propeller. The drone chassis multibody masses in Figure 13 follow the mathematical model in Equation 12. Each inertia tensor can be defined individually in this system, which reduces to Equation 13.

$$\tau = I_{XYZ} * \alpha = \begin{bmatrix} I_{XX} & I_{XY} & I_{XZ} \\ I_{YX} & I_{YY} & I_{YZ} \\ I_{ZX} & I_{ZY} & I_{ZZ} \end{bmatrix} \begin{bmatrix} \dot{\omega}_X \\ \dot{\omega}_Y \\ \dot{\omega}_Z \end{bmatrix} \qquad (12)$$

$$\tau = \begin{bmatrix} I_{XX} & 0 & 0 \\ 0 & I_{YY} & 0 \\ 0 & 0 & I_{ZZ} \end{bmatrix} \begin{bmatrix} \dot{\omega}_X \\ \dot{\omega}_Y \\ \dot{\omega}_Z \end{bmatrix} \qquad (13)$$

Rotor: The **Rotor** package contains the rotor models, as shown in Figure 14. The rotor is linked mechanically to the motor, airframe, and blades. The multibody connectors labeled `torque_1`, `torque_2`, and `force` are connected to the `torque_1`, `torque_2`, and `thrust_out` connectors respectively on the motor in Figure 16. This links the torque from the machine to the revolute. The speed at which the revolute is turning is determined by a scaled measurement of the relative angular velocity between `torque_1` and `torque_2`.

The aerodynamic forces are applied using the ω^2 model in Equations 14 and 15. Equation 15 is used for the aerodynamic torque is calculated here, and the thrust is calculated in the component `aero_torque`. The thrusts are coupled to the motor component using the multibody connector `force`. The thrust is calculated in Figure 16 using the real expression block `thrust`, which uses Equation 15 and 16

$$\tau_h = 0.0015\omega^2 \qquad (14)$$

$$\tau_o = (3.5 \times 10^{-6})\omega^2 \qquad (15)$$

$$P_p = \tau\omega \qquad (16)$$

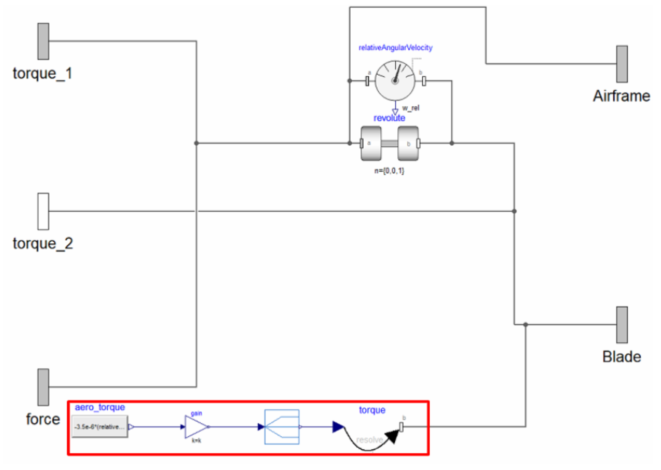


Figure 14. Rotor model in Dymola. The red box in the lower left corner of the model represents the calculations needed to determine aerodynamic forces applied to the rotor.

Motor: The machines included in the **Machines** subpackage are configured to link the machines to the controller and revolute. The machine output is controlled by a `real` signal from the controller, as shown in Figure 5. In Figure 16, the machine output is represented in terms of rotational connections, so the torque sensor, force, and torque components create the multibody connections necessary to link to the rotor. The `gain1` component in Figure 16 adjusts the direction the motor turns, where a gain of "1" turns the motor clockwise and "-1" turns the motor counterclockwise. The motor also contains a speed controller that uses the position signal from the main controller and the battery state of charge. This controls the voltage supplied to the motor and scales it to the available battery voltage, which is connected with electrical pin `p1`.

Examples and Test Systems

All of the components described are configured to create the simple drone model shown in Figures 17 and 5. Figure 17 shows the drone with an ideal power source, while Figure 5 shows the drone when a battery is added to the system. The system in Figure 5 also has the `frame_a1` connector that can be linked to additional external payloads, such as a camera. The drone is tested in models configured in the **Examples** package. These examples include using different input signals to control the inputs `xcoord`, `ycoord`, and `zcoord`. These input signals can be from the signals provided in the library, signals from the Modelica Standard Library, experimental data, and custom signal functions defined by the user. The inputs for the model can also be left disconnected from any inputs and compiled as a Functional Mock-Up unit (FMU). By selecting this option, the model can be exported to other software tools for analysis and simulation.

The model tests for visualization and VR interaction are saved in the **Tests** package. These models are developed from the same components in the library previously described with the

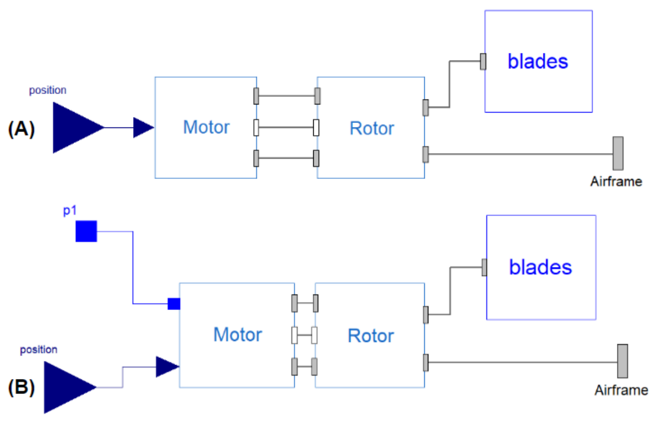


Figure 15. Model for the propeller system containing mechanically linked motor, rotor, and blades (A) without; (B) with power connections.

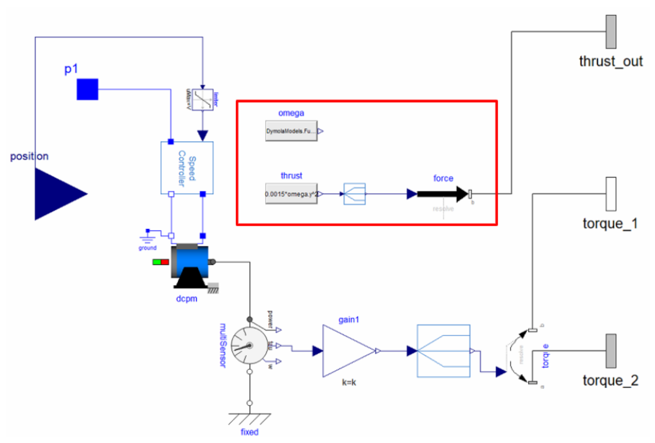


Figure 16. Model for the motor in the propeller system containing a DC motor applying force to the propeller blades. The red box contains the components used to calculate the thrust.

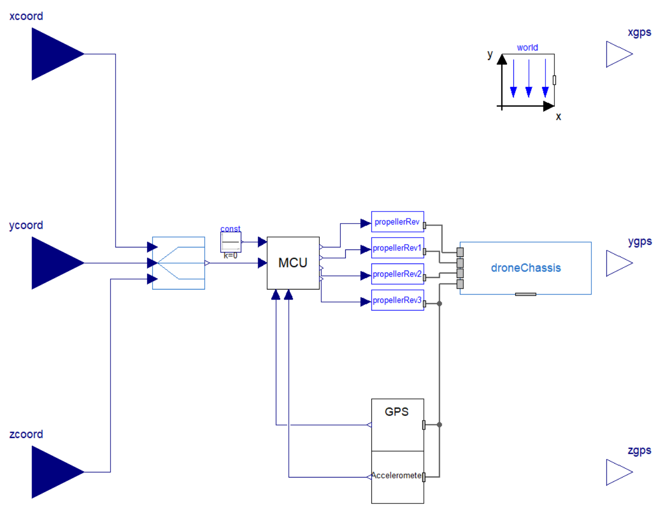


Figure 17. Complete drone model consisting of propellers, motor, controller, and chassis with ideal power system. Inputs come from x, y, and z coordinate location.

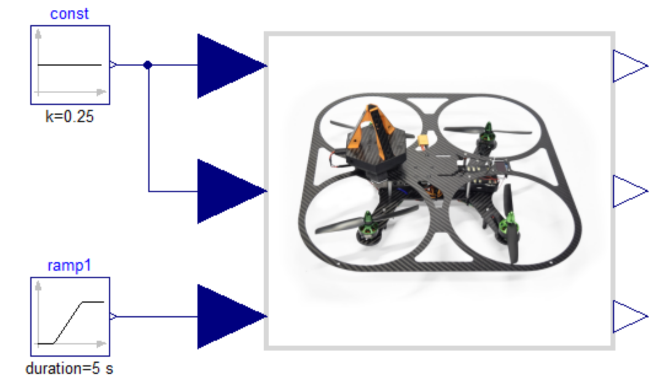


Figure 18. Drone model configured for 1 m/s 5 second ramp input in the Z direction test.

ability to simulate and animate objects from CAD files and other pre-defined shapes using the DLR Visualization library (Ref. 14). These conditions are tested using the models saved in the **Examples** package. They are controlled using a ramp signal in the Z-direction to linearly move the drone up to a height of 5m while fixing the X and Y coordinates to zero, which is shown in Figure 18.

The `world` component in Figures 5 and 17 applies the gravity field to every multi-body component in the negative Z direction. It is not connected to any of the bodies since the MSL multibody library package has been configured to propagate gravitational parameters into each component.

DRONE VISUALIZATION AND ANIMATION

When the drone is simulated, the behavior can be observed as an animation. The drone has been configured to use 3D Object (.STL) files to represent the propellers and body of the drone in an animation of the drone, which appears when the drone is simulated. The 3D Object files are defined in the chassis and blade models in Figures 11 and 13 as `fixedShape` components.

The initial position of the drone is shown in Figure 19. The propellers move over time, as shown in Figure 20. These snapshots were taken while the drone was hovering a height of 5 m, where the drone oscillates slightly in the Z-direction while spinning the propellers. This oscillation is shown in Figure 22 occurring between 5 and 10 seconds.

The propellers spinning are also shown in Figure 21, where the drone is steadily moving to a height of 5 m over a 5 second period, following the same flight path outlined in Figure 18. The trace of the propeller shows that the drone oscillates in the Z-direction due to the dynamic response of the controller, as discussed in a study below.

STUDIES

The developed library contains models for varying degrees of complexity for the electric power system. The lowest level of complexity might include the ideal version of the component, while the most complex models consider losses and non-ideal

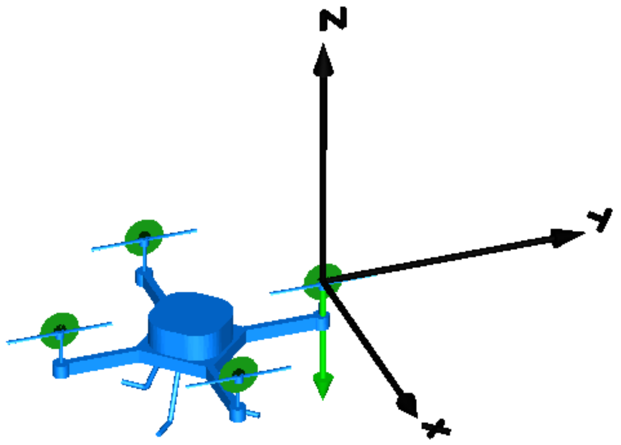


Figure 19. Drone animation at time = 0 seconds.

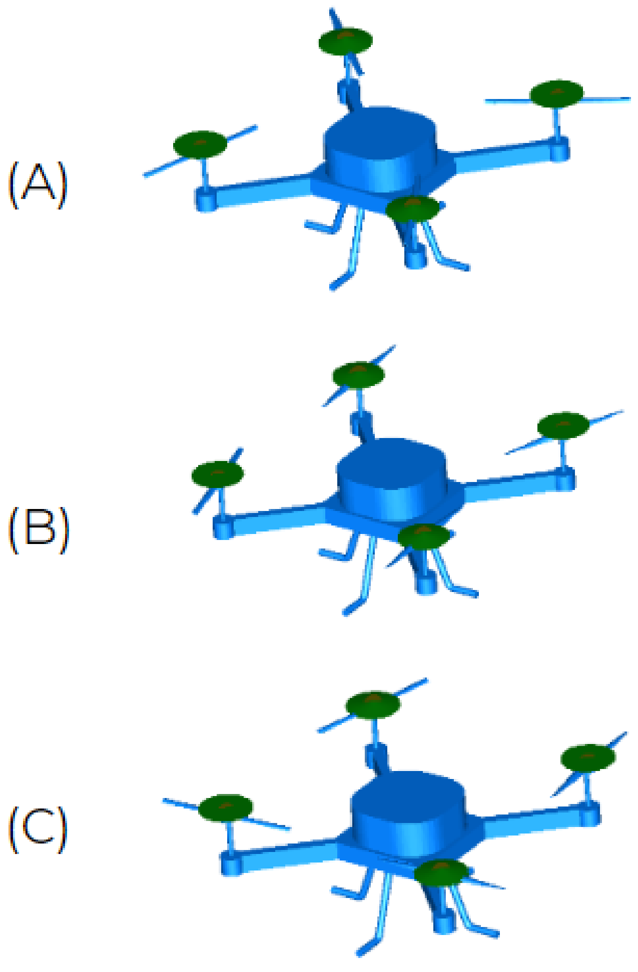


Figure 20. Drone animation at time (A) 1 seconds at 1 m (B) 1.5 seconds at 1.5 m (C) 2 seconds at 2 m for an ideal motor and ideal power system.

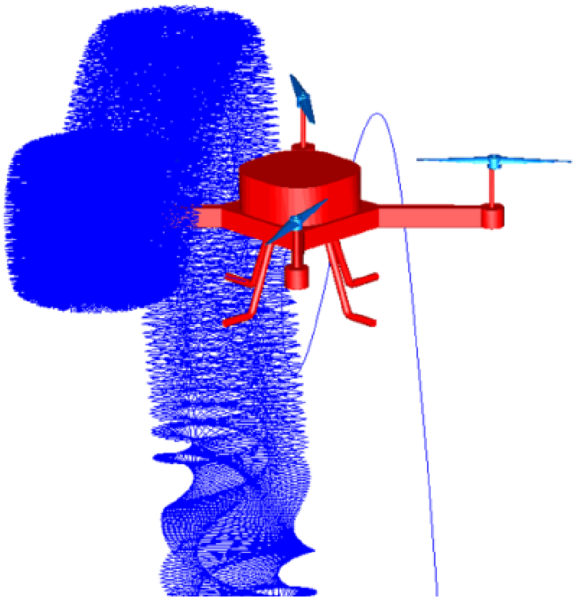


Figure 21. Drone animation with the path of the propeller shown as a trace for the flight path in Figure 18.

behavior. In this case, the drone is tested using an ideal power system with a first order motor model, a DC machine with losses and an ideal power system, a battery and DC machine with losses, and the battery with a converter and DC machine with losses to demonstrate the effects of the modeling complexity of each of the components on the drone operation. This helps in determining the electric power requirements that will arise from different operating conditions. In this section, several analysis are made with the tests summarized in Table 1. The columns refer to the scope of the study, and the rows show the model variants of the power system used for each study.

Table 1. Study scope and power system model variants.

	Ideal Flight Path	Payload
Ideal motor, ideal power system	•	•
DC motor, ideal power system	•	•
DC motor, battery without converter	•	•
DC motor, battery with converter	•	•

IDEAL FLIGHT PATH REFERENCE TRACKING

Ideal Power System, Ideal Motor

The simplest representation of the drone is modeled with an ideal motor with a ideal voltage source power system. This assumes that the voltage and current supplied to the system is constant and can consistently meet the needs of the system. The entire system modeled with the ideal components is shown in Figure 5.

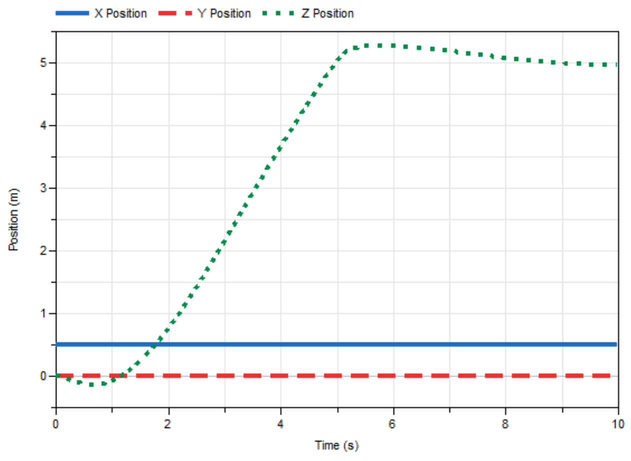


Figure 22. Drone XYZ position under a ramp signal input for an ideal motor and ideal power system.

Figure 22 shows this model is simulated under ideal conditions when the Z position of the controller is subject to a ramp signal from the ground to 5 m. The drone overshoots the position in the Z direction when the drone reaches its final hovering point by 5.43%. The low damping in the ideal motors also causes the oscillation in the X and Y direction. Figure 24 shows the effect of the motor damping on the force applied to the propeller blades in the Z direction. The damping constant in the motor with losses is higher than the ideal case, which is why these is some oscillation at the beginning of the simulation as well as a larger force to stabilize the system during the ramping period. The oscillations in the beginning of the simulation for the DC motor is also due to the charging of the inductances in the motor. The drone is modeled to take the gyroscopic position into account when determining the XYZ coordinate position of the drone (as shown in the controller in Figure 9), so the oscillation causes the change in position in the X and Y direction.

Ideal Power System, Permanent Magnet DC machine with Losses

The simplified DC machine model described in the previous section is replaced with a permanent magnet DC machine from the MSL (Ref. 9). It is assumed that an ideal lossless power supply holds the voltage at a constant value and can provide the necessary current throughout the duration of the test.

When the drone is placed under the same operating conditions as before, it shows better stability in the X and Y direction and overshoots less when the drone reaches the hovering height of 5 m. The position of the drone over time is shown in Figure 23. The DC machine provides more damping to the system than in the case of the ideal motor, so the drone does not overshoot position as much in the Z position when reaching the hovering height and it doesn't cause the drone to move in the X and Y direction; in fact the system is well damped when the permanent magnet DC motors are used in the system. Figure 24 compares the force applied to the blades from the DC and

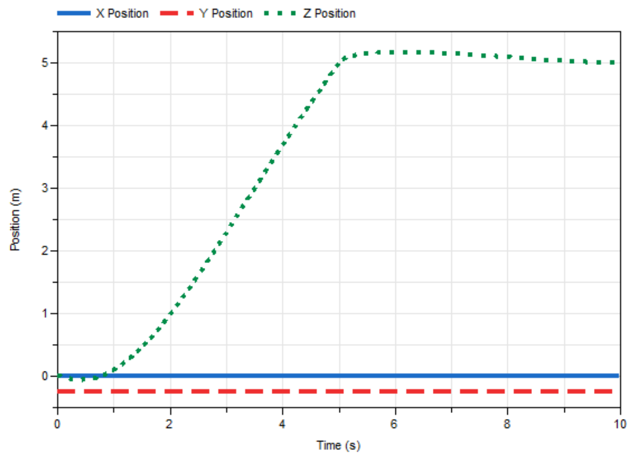


Figure 23. Drone with DC permanent magnet electrical motor and ideal power system XYZ position under a ramp signal input.

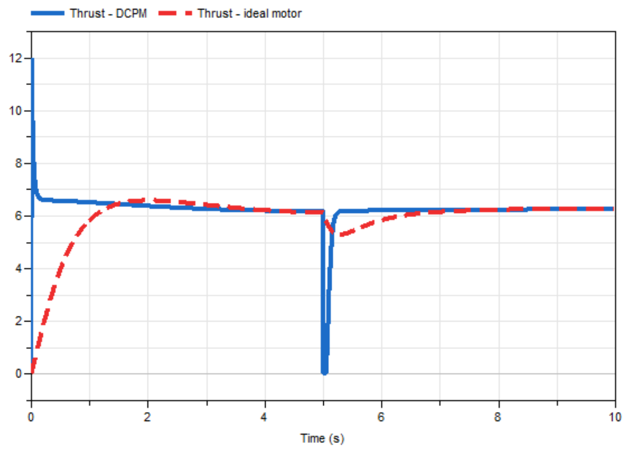


Figure 24. Thrust from one motor in the Z direction compared for the ideal and non-ideal machine.

ideal motors. By increasing the model detail of the motor, the force applied to the blades increases due to the increased damping.

Non-ideal Power System, Permanent Magnet DC machine with Losses

The permanent magnet DC machine from the MSL and the battery power system described in the **Sources** section are used to model the drone. The DC machine has a nominal voltage of 12V. The battery starts at a voltage of 9.55V with a state of charge (SoC) of 0.6. When the drone is tested under the same conditions as before, the discharging battery does not affect the behavior, i.e. the system response is the same as before. In this case, the battery assumes a constant power consumption rate, when in reality it discharges at a variable rate depending on the operational state of the drone.

Figure 25 shows the flight path of the drone subject to the same operating conditions shown in Figure 18. The position of the drone is similar to the response when an ideal power

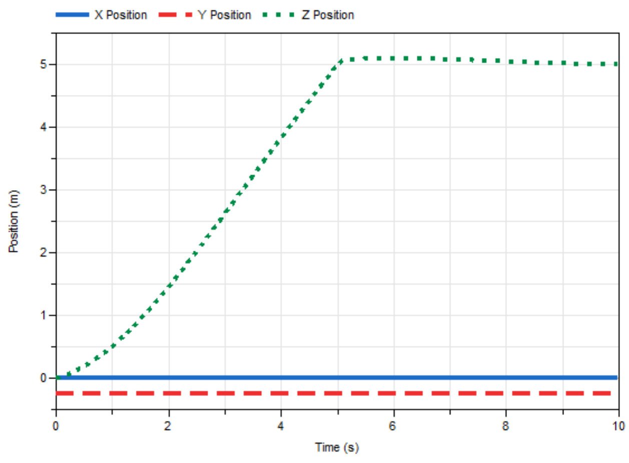


Figure 25. Drone with DC permanent magnet electrical motor and battery power system XYZ position under a ramp signal input.

system is used, except the changing voltage in the system cause the drone to move slightly in the X and Y direction. The drone moves faster in the X position than when the ideal voltage source is used.

The battery discharges at a constant rate of 0.004 percent per second , which causes the voltage to step down at a rate of 1e-4 V per second. Figure 26 shows the battery's state of charge and voltage decreasing over the 10 second simulation period.

Non-ideal Power System, Permanent Magnet DC machine with Losses and DCDC Converter

The controller is now replaced to have an input from a DC to DC step down converter as per the model in Figure 5. The system response is shown in Figure 27. The time domain specifications and energy consumption for the motor and controller are shown in Table 3. The DCDC step down converter creates a large voltage ripple, but the system stays stable. This is explained in the "Power system impact on closed loop dynamics" section.

POWER SYSTEM IMPACT ON CLOSED LOOP DYNAMICS

When the drone is modeled to have a controller that requires a power input, a DC to DC step down converter is required to step down the 12.1 V from the battery to 5V to be used by the controller. The DC to DC controller in Figure 8 has transistors and diodes that, depending on how the transistors are modeled, will cause different levels of voltage ripple. This voltage ripple is fed back into the motor voltage input. The power system for the Otus quadcopter in Figure 4 does not have a voltage controller for the motors. The voltage profile in Figure 29 shows the ripple from the DCDC converter affecting the battery voltage. As the system stabilizes, the voltage ripple decreases. This voltage ripple propagates through to the scaling of the position signal.

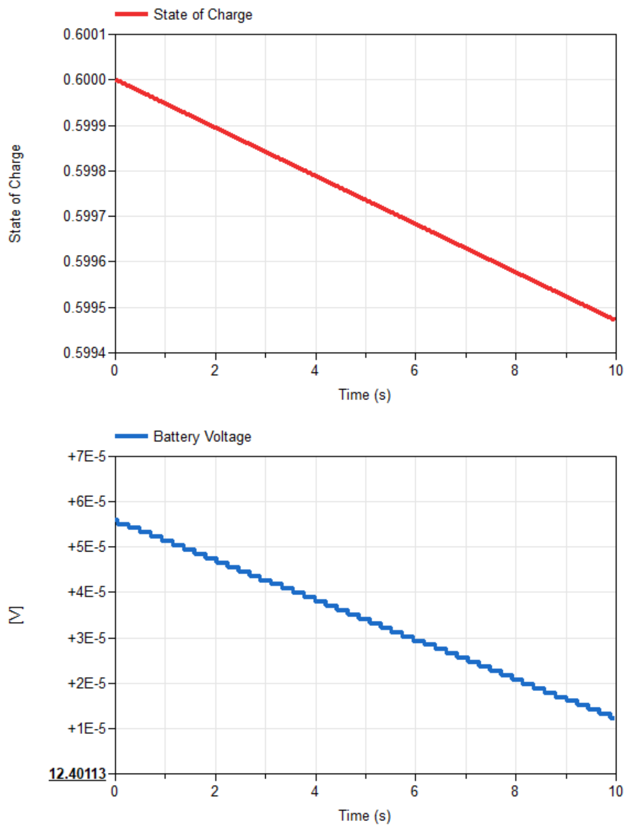


Figure 26. Drone with DC permanent magnet electrical motor and battery power system battery state of charge (top) and voltage (bottom) under a ramp signal input.

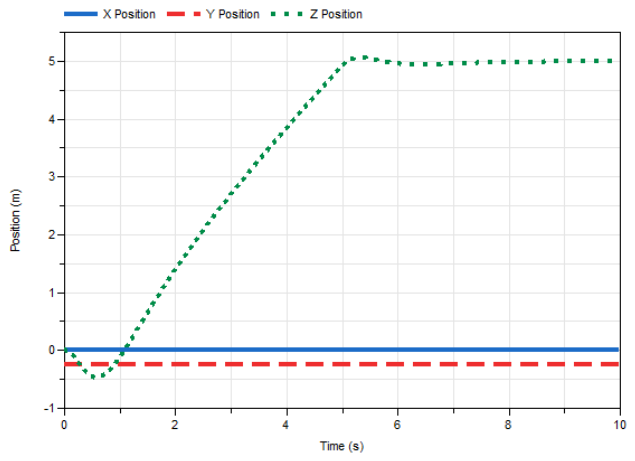


Figure 27. Drone with DC permanent magnet electrical motor, battery power system, and DC to DC converter XYZ position under a ramp signal input.

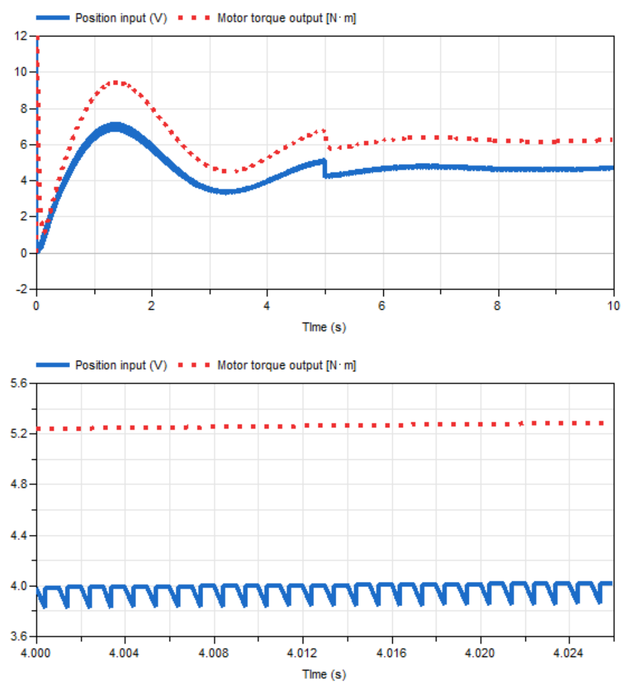


Figure 28. Position signal compared to motor output when converter is connected to the power system and supplies voltage ripple.

As described in the **Controllers** section, the speed controller uses the battery voltage and current to adjust the position signal from the controller accordingly. Figure 28 shows the position input compared to the machine torque output. The solid line in the figure shows the position input to the machine measured in V, which has been scaled by the speed controller to consider the battery voltage. The battery ripple in Figure 29 is evident in the input voltage to the machine. The dotted line in Figure 28 shows the mechanical shaft torque output by the machine. The torque has the similar characteristics as the input voltage, but the output torque does not have the same ripple. The inductances inside of the machine damp the output torque, largely eliminating the ripple in the output.

Another test was run to determine the point at which the battery cannot allow the drone to fly anymore. Using the same simulation set up of ramping the drone to a height of 5m and continuously hovering for 13 minutes. Figure 30 shows the discharge of the battery over the discharge period. When the battery discharges to a point of about 8V, the drone falls down to the ground.

PAYLOAD VARIATION

One of the attractive uses of UAV is the possibility to carry different payloads. However, these can impact both the system dynamics, the power systems performance, and energy consumption. This section aims to address some of these aspects.

As outlined in Figure 5, the drone model has a rotational connection (labeled as `frame_a1`) to connect payloads onto the

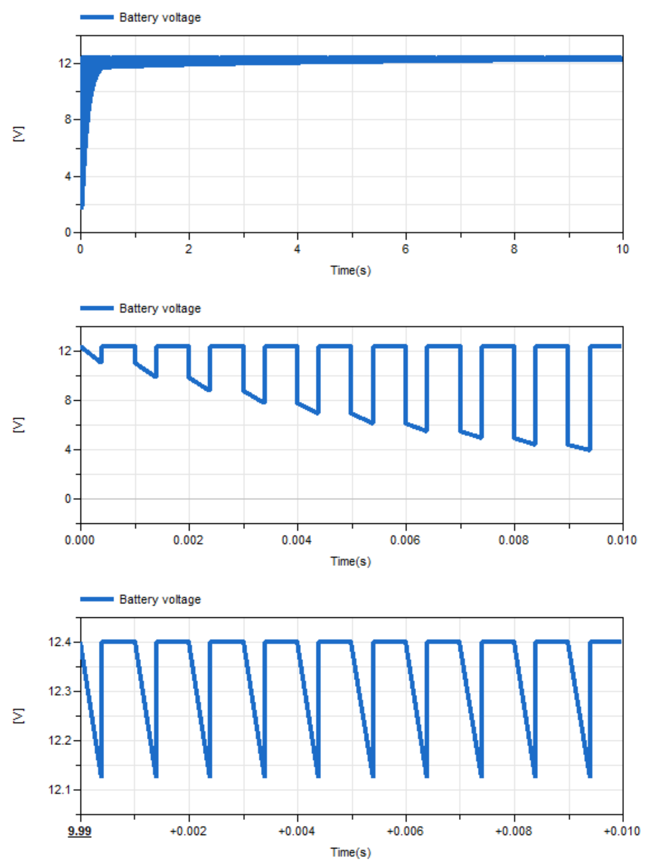


Figure 29. Battery voltage with voltage ripple from DCDC converter. The top shows the voltage over the entire 10 second simulation period, the middle and bottom shows a the same signal, but magnified to show the ripple.

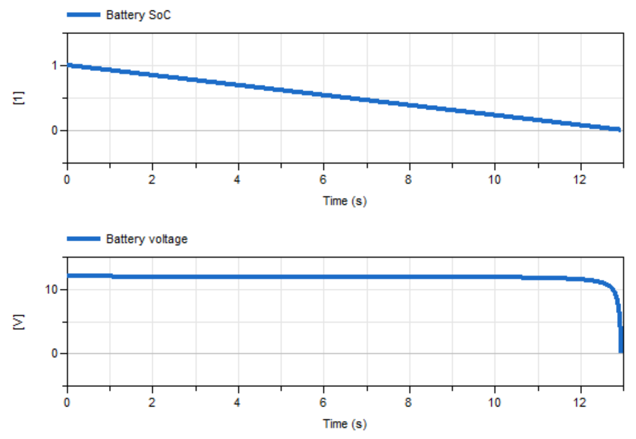


Figure 30. Battery voltage and state of charge over 13 minutes at 5m hovering.

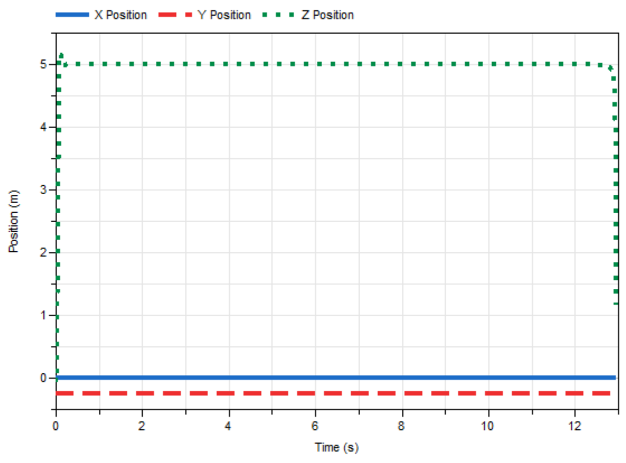


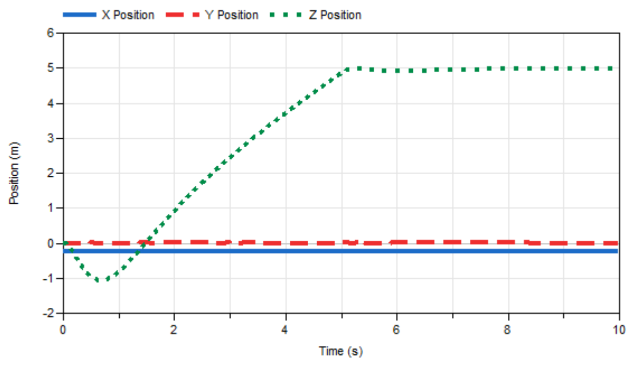
Figure 31. Drone altitude over the entire discharge test. The top figure shows how the drone altitude, while the middle and bottom figures zoom in on the point of simulation failure.

rotorcraft. In this test, the drone is attached to a 3kg payload in the Z direction, effectively being modeled as a mass attached to the airframe. The ideal flight path used in the reference tracking tests is applied to the drone with the payload. Figure 32 shows the drone position with the 3kg payload attached. In this case, the drone is not initialized prior to adding the payload or applying the flight path. The drone overshoots its final hovering height by 1.86%. It also has a steady state error of -1.64e-4%. As outlined in Table 2, the motors in the drone consume more power when the payload is attached. Similar to the tests without the payloads, the case of the drone using an ideal power source results in a larger energy consumption by the motors. Similar to the other reference tracking tests outlined in Table 2, the power consumption of the motors is large for the ideal case, but as details are added to the model, the power consumption decreases. For brevity, the tracking response plots of all configurations under a payload are not included in the paper.

Comparing cases 5, 6, and 7 from Table 2 in the Appendix, it can be noticed that the addition of the payload leads to a power increase of approximately 2000 W (case 2 vs. case 6, case 3 vs. case 7, case 4 vs. case 8). This will limit the potential flight duration or increase the power requirements from the source (and hence increase the weight). Comparing cases 6 and 7 indicates that substantial power consumption takes place when stabilizing the drone with the additional payload, about 35 W, which is about 80% additional power than the power consumed without the payload. Similarly in configurations, such as comparing cases 2 and 8 from Table 2 in the Appendix, the 3 kg payload increases power consumption by about 200 W.

CONCLUSIONS AND FUTURE WORK

The drone model presented in this paper provides a basis for open-source, multi-domain drone modeling at different levels of model complexity. These models are more complex than



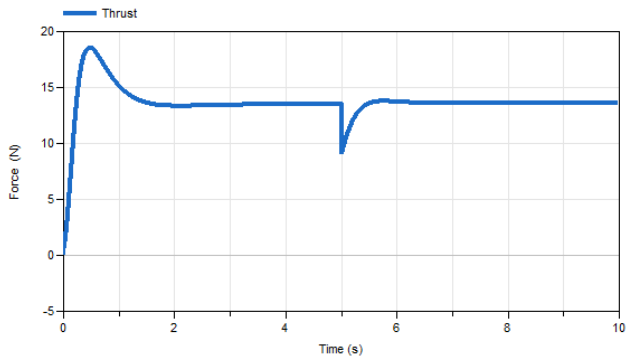


Figure 32. Drone position with a payload of 3kg applied to airframe (top) and thrust per rotor (bottom).

previous models developed in the literature, and in addition, they allow for animation the drone for a given input, which is beneficial for insight, analysis, and communication between domain specialists.

Future versions of the drone model will increase the complexity of the drone's representation with better aerodynamics. This includes modeling heating losses of the electrical and mechanical components in the system. Thermal behaviors are included as an option to model the losses of the electrical and mechanical components in the system, such as heat dissipation from the resistors and friction, but they currently not used in the drone model and will be included in future versions of the model. This can serve to illustrate the importance of thermal management when fixed power supply is used (i.e. batteries). Averaged power electronic models will illustrate the impact of averaged modeling on the drone and the impact on heating, and the electrical and closed loop response. Additional electrical loads that will be attached to the power system as a payload, such as cameras, will also be included in future iterations of the model. The speed controller model will also be improved to consider a nonlinear relationship between battery voltage and motor power.

Collision detection will also be added to future versions of the model. In the reference tracking studies, the drone falls into the -Z plane, which is physically unrealistic. A collision detection system (Ref. 15) (Ref. 14) will prevent this from happening as well as provide realistic operation when the drone in used in VR simulation and interacts with the provided environment. Penalty-based collision techniques and the Bullet Physics Library will be applied to implement collision detection according to the methods described in (Ref. 15). The Idealized Contact Library (Ref. 16) (Ref. 17) will also be explored to model and handle collision simulations.

Experimental data from a gyroscope (Ref. 11) test bed being developed at RPI will be used for our quadcopter simulations in future studies; these results will verify that the drone model in the library will be able to reproduce the behavior of real aircraft. It will also provide a basis for system identification studies of cyber-physical system in the future. This will also allow for the calibration of the control system. The animation models will also be exported to a virtual reality environment to allow for interaction with the drone.

The model source files are available at: https://github.com/ALSETLab/Modelica-Drone-3D-FMI.

Author contact: Meaghan Podlaski (podlam@rpi.edu), Luigi Vanfretti (vanfrl@rpi.edu), Hamed Nademi (nademh@rpi.edu), Hao Chang (samariumch@gmail.com)

ACKNOWLEDGMENTS

This work was supported in whole or in part by the National Aeronautics and Space Administration through the University Leadership Initiative award for the Center for Cryogenic High-Efficiency Electrical Technologies for Aircraft (CHEETA), in part by Dominion Energy, in part by the ECSE Department at RPI, and in part by the Center of Excellence for NEOM Research at King Abdullah University of Science and Technolog. The first author is supported through a Rensselaer Graduate Fellowship.

The authors would like to thank Dr. Farhan Gandhi and Dr. Robert Niemiec for their invaluable input in shaping the work presented in this paper.

REFERENCES

1. Bresciani, T., “Modelling, Identification and Control of a Quadrotor Helicopter,” MSc Thesis 166609, October 2008.

2. Kuric, M., Osmic, N., and Tahirovic, A., “Multirotor Aerial Vehicle modeling in Modelica,” , 07 2017. DOI: 10.3384/ecp17132373

3. X. Ma, Z. L., and Nae, C., “Multi-domain modeling and Simulation of a Quad-rotor aircraft based on Modelica,” 2016 Int’l Conf. Modeling, Sim. and Vis. Methods, 2016.

4. Luukkonen, T., “Modelling and control of quadcopter,” Technical report, August 2011.

5. Mahony, R., Kumar, V., and Corke, P., “Multirotor Aerial Vehicles: Modeling, Estimation, and Control of Quadrotor,” *IEEE Robotics Automation Magazine*, Vol. 19, (3), Sep. 2012, pp. 20–32. DOI: 10.1109/MRA.2012.2206474

6. ALSETLab, “Modelica Drone 3D FMI,” https://github.com/ALSETLab/Modelica-Drone-3D-FMI/, October 2019.

7. DLR, “Modelica Synchronous Library,” https://github.com/modelica/Modelica_Synchronous, February 2020.

8. Modelon, “Modelon Library Suite,” https://www.modelon.com/products-services/modelon-library-suite-modelica-libraries/, February 2020.

9. Association, M., “Modelica Standard Library,” https://github.com/modelica/ModelicaStandardLibrary, February 2020.

10. Benchmark, R., “Otus Quadcopter Manual V1.1,” https://rcbenchmark.gitlab.io/docs/en/download.html, February 2020.

11. Dynamics’, E., “FFT Gyro,” https://eurekadynamics.com/fft-gyro/, October 2019.

12. Li, S., and Ke, B., “Study of battery modeling using mathematical and circuit oriented approaches,” 2011 IEEE Power and Energy Society General Meeting, July 2011. DOI: 10.1109/PES.2011.6039230

13. Otter, M., Elmqvist, H., and Mattsson, S. E., “The New Modelica MultiBody Library,” , 2003.

14. Hellerer, M., Bellmann, T., and Schlegel, F., “The DLR Visualization Library - Recent development and applications,” Linköping Electronic Conference Proceedings, The 10th International Modelica Conference 2014, March 2014.

15. Hofmann, A., Mikelsons, L., Gubsch, I., and Schubert, C., “Simulating Collisions within the Modelica MultiBody library,” , 03 2014. DOI: 10.3384/ecp14096949

16. Oestersötebier, F., Wang, P., and Trächtler, A., “A Modelica Contact Library for Idealized Simulation of Independently Defined Contact Surfaces,” , 2014.

17. Oestersötebier, F., Wang, P., and Trächtler, A., “Idealized Contact Modelica Library,” https://github.com/oestersoetebier/IdealizedContact, March 2020.

APPENDIX

Table 2. Energy metrics for the ideal flight path tests.

#	Test type	Power consumption, total (motor) (Joules)	Power consumption, during ramp (motor) (Joules)	Power consumption, during hover (motor) (Joules)
1	Ideal motor, ideal power system	4640.3	2387	2253.3
2	DC motor, ideal power system	1661.5	851.09	810.41
3	DC motor, battery without converter	1661.4	850.9589	810.4825
4	DC motor, battery with converter	1659.5	845.30	814.22
5	Ideal motor, ideal power system, and 3kg payload	12500	6825	5675
6	DC motor, ideal power system, and 3kg payload	3628.1	1830.1	1798
7	DC motor, battery without converter and 3kg payload	3629.1	1832.3	1797.0
8	DC motor, battery with converter and 3kg payload	3614.1	1830.1	1797.1

Footnote: The energy consumed by the motor controller is a constant 0.25 J.

Table 3. Tracking response for the ideal flight path tests.

#	Test type	Overshoot	Steady state error (Z)
1	Ideal motor, ideal power system	5.43%	0.054%
2	DC motor, ideal power system	0%	0.021%
3	DC motor, battery without converter	1.36%	0.13%
4	DC motor, battery with converter	1.04%	-0.072%
5	Ideal motor, ideal power system, and 3kg payload	17.6%	-4.48%
6	DC motor, ideal power system, and 3kg payload	1.86%	-1.64e-4%
7	DC motor, battery without converter and 3kg payload	1.04%	0.2%
8	DC motor, battery with converter and 3kg payload	-0.88%	-0.15%