Simulating Quadrotor Dynamics using Imported CAD Data

Ryan F. Gordon[1], Preethi Kumar[2] and Richard Ruff[3]
Mathworks Inc, Natick,MA,01760

Quadrotors have grown in popularity with the increased availability of low-cost hardware. Low-cost platforms enable students, hobbyists and research engineers to design and fly custom software on a limited budget. As part of this design process, modeling and simulation can be used to improve the flight characteristics of the vehicle and enable more complex control designs. Common design patterns in aerospace system design can be applied to the quadrotor design to improve the development process. This paper describes the start of the design and development process — the transition from CAD data to a full dynamic model.

Nomenclature

PWM	=	*pulse width modulation*
CAD	=	*computer aided design*
T	=	*propeller thrust*
I	=	*propeller index*
L	=	*distance from center of quadrotor to center of propeller*
z_b	=	*body frame z-position*
w	=	*wind velocity*
C	=	*rotation direction of propeller*
S	=	*rotation rate axis of propeller*
k_t	=	*coefficient of thrust*
ρ	=	*air density*
A_p	=	*propeller area*
α_i	=	*propeller rotational rate*
P	=	*propeller pitch*
R_p	=	*propeller radius*

I. Introduction

Studies of quadrotor mechanics typically focus on a first-principles approach in which the vehicle's equations of motion are defined to determine how the forces and moments of the quadrotor are applied to a dynamic model. With advanced modeling and simulation technology it is possible to instead define the dynamics of the system via the topography of the design and apply the propeller forces directly. This approach simplifies the modeling process because the designer no longer needs to derive the equations of motion of the system. To further simplify this process, the designer can automatically generate the topography of the mechanical system by importing CAD data into the simulation software. Since CAD models are often developed in tandem with the simulation, importing the CAD design can significantly speed up the process of building and updating the plant model used in simulation studies. This process can also reduce the number of errors introduced when design changes occur since they are implemented systematically instead of by hand.

In this paper we extend a quadrotor case study[1] previously presented by using CAD data, obtained from the quadrotor supplier, to construct a higher fidelity plant model. This CAD data is converted into XML data that can be read by a physical modeling tool. Once in this domain the system dynamics can be analyzed and aerodynamics added. Furthermore, the model enables the control design process to begin before the vehicle design is completed. Once the vehicle is available, flight code can be generated and tested almost immediately.

---

[1] Product Marketing Manager, Simulink Platform Marketing, MathWorks, 3 Apple Hill Dr. Natick, MA 01760.
[2] Controls Tools Developer, Language of Technical Computing, MathWorks, 3 Apple Hill Dr. Natick, MA 01760.
[3] Principal Application Engineer, Application Engineering, MathWorks, 3 Apple Hill Dr. Natick, MA 01760.

II. Importing Data from 3D CAD Software

On large projects the development of one team is often dependent on the results of another. In this case study the modeling, simulation, and control of the quadrotor sits in the middle of two major development efforts. First, a supplier designed the vehicle and defined the geometry for the control design team. After the team designs the simulation and control system, the final effort will involve generating flight code and deploying to the vehicle. In this example, the geometry is primarily defined in a SolidWorks assembly. This data can be imported into SimMechanics®[4], a physical modeling tool, to simulate the dynamics of the system with Simulink®[5] and begin the control design process.

With Model-Based Design[6], engineering teams can complete the design and test of the control system prior to acquiring the final hardware from the manufacturer. In this case, the team is able to design the control software in parallel with the physical hardware manufacture. Using these methods the team can significantly accelerate their development of the flight control system as they do not need to wait for the supplier to finalize and deliver the hardware. The process of creating the model, performing simulations to study the dynamics, and designing the initial flight controls for this system is described in more detail throughout the sections of this paper.

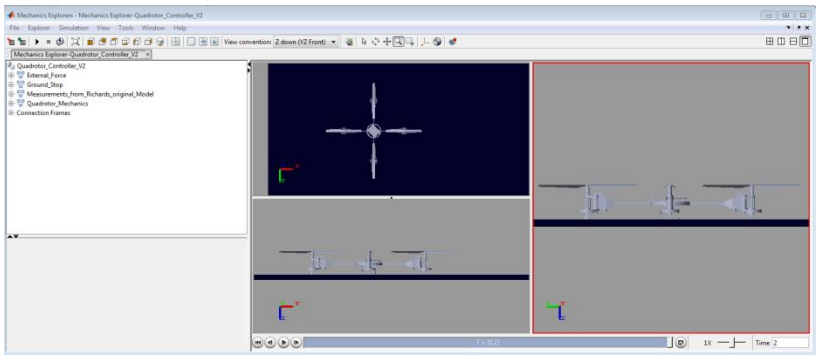


Figure 1 Mechanics Explorer View of Quadrotor

The steps involved in importing a CAD model into SimMechanics are:

1. Install and register SimMechanics Link®

   SimMechanics Link® is a utility that needs to be installed and registered using a MATLAB® session. This process makes SimMechanics Link® available in the CAD platform as an Add-In tool that enables exporting CAD assemblies.

2. Export the CAD model to an XML file

   Using the Add-in tool, the CAD model can be converted to an XML file that contains a description of the model's geometric properties in a format that SimMechanics® can analyze.

3. Import the XML file into SimMechanics®

   This XML file can then be imported to SimMechanics® using a simple command ('smimport'). The resulting Simulink® model has the same geometric properties (including dimensions, position, orientation, and inertia) as those defined in CAD.

III. Improving Simulation and Evaluating Dynamics

A. Connecting Physical Components with Joints

Through the import process, we specify the geometric properties of the quadrotor (figure 2), as provided by the quadrotor supplier. The next step is to define the quadrotor motion, which is done it two steps:

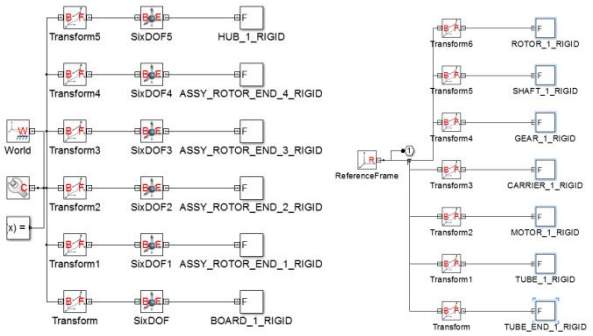


Figure 2 Initial Mechanical Design after Import

motor-propeller pair. On simulating the model with the revolute joints, the Mechanics Explorer tool allows us to verify the motion occurs on the desired axis.

To define the force produced due to this motion we actuate the revolute joints with external forces. The amount of thrust generated by the propellers is a function of the rate of rotation of the propellers and the angular rate of rotation of the body along the roll and the pitch axes. On simulating this setup without power supplied to the motors, the quadrotor can be seen falling due to the influence of gravity.

1. Define how the propellers move with respect to the motor (figure 3).

2. Define the amount of thrust produced by this movement.

To define the motion of the propellers with respect to the motors, we use four revolute joints, one between each

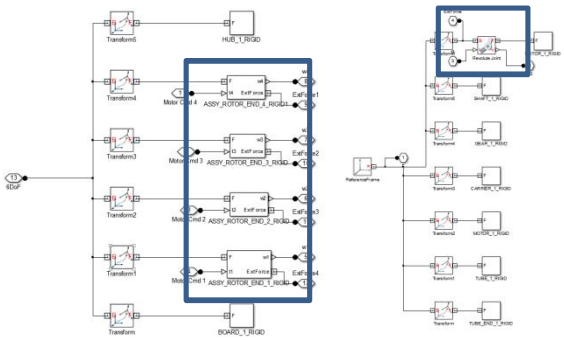


Figure 3 Modified Design After Import

The thrust can be defined by building up physical signals, which are part of the physical modeling environment, based on the thrust equation below. Applying the resultant thrust to the revolute joint will allow each propeller to generate the appropriate thrust based on it's rotational rate.. The thrust can be defined as[2]:

$$T_i = C_i\left(\frac{1 - 2\pi LCS}{P\alpha_i} + 2\pi\frac{z_b - \dot{w}_{z_b}}{P\alpha_i}\right)$$

Where C is defined:

$$C_i = k_t \rho A_p \alpha_i^2 R_p^2$$

With most of the constants defined by the geometry of the vehicle and its dynamics, we assume reasonable values for the thrust constant, k_t, until it can be determined from testing the motors and propellers of the actual vehicle. Once this is accomplished we fine tune the controller gains, deploy the control system on the target hardware, and flight test the vehicle to refine the model.

B. Simulating a Physical Ground

To more closely simulate a real world application, we add a ground stop block to the model at this point. This ensures that the vehicle does not fall infinitely. Essentially, this simulates hard ground from which the quadrotor can take off and land. Currently, the ground is assumed to be a flat surface. In the future, modifications can be made to this block to simulate a sloped or uneven surface.

C. Adding Motor Dynamics

Next, we need to provide input to the motors that will control the maneuvers performed by the quadrotor. The plant model consists of the quadrotor's frame, the

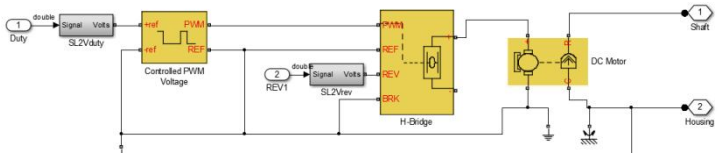


Figure 4 DC Motor Modeled with Physical Modeling Components

four motors (figure 4), the four propellers, and the electronics required to navigate the vehicle. Four electronic speed controllers (ESC) regulate the speed of the four motors according to the input PWM signal. The input to the speed controller is a voltage, which is indicative of the duty-cycle of the PWM signal that drives the ESC. Thus, we add blocks to the model that take a voltage input and produce a PWM signal that can be used to drive the motor. A power amplifier with a controlled PWM voltage generator and an H-bridge serves as the ESC. At this point, the simulation is still open loop. The H-bridge enables the two motors (placed opposite each other in the quadrotor's frame) to rotate in the clockwise direction, while the other two rotate in the counterclockwise direction. This step allows the moments about the opposite pairs of motors to cancel out, preventing the quadrotor from spinning about its center (yaw-axis) continuously. The same positive voltage is then fed to all four motors to test the simulation at this stage. On providing a sufficiently high voltage, the quadrotor will produce enough thrust to overcome gravity and lift off.

IV. Preliminary Control Design and Model Integration

The final step is to design a controller that stabilizes the plant and supports tracking. Given a target altitude, pitch and roll, the controller will produce the required motor voltage inputs. Three cascaded loops form the control architecture. Each loop consists of an outer rate controller and an inner position controller. Each controller has a proportional, integral, and a derivative term. We use the PID blocks available in Simulink and can be tuned with Simulink Control Design[3]. Each PID block is associated with a PID tuner, which can be used to specify a desired response time and the desired transient behavior.

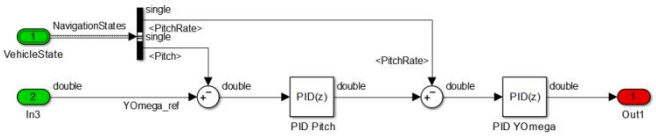


Figure 5 Preliminary Control Loops on Position and Rate in Three Axes

The PID tuner works by first linearizing the plant model around the controller block. The algorithm then determines an initial guess of appropriate gains based on the results of the linearization. Finally, the PID tuner interface provides a set of sliders that we use to achieve the desired response. Typical PID objectives include:

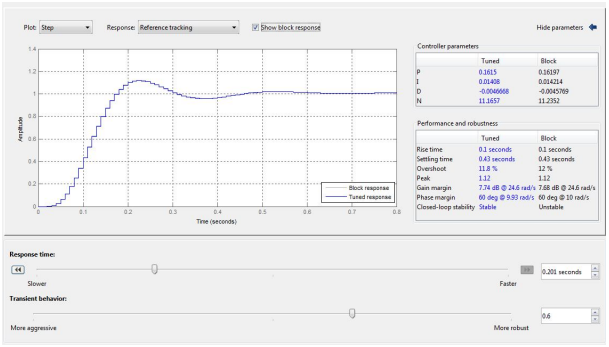


Figure 6 PID Tuner Interface

- Closed-loop stability — The closed-loop system output remains bounded for bounded input.
- Adequate performance — The closed-loop system tracks reference changes and suppresses disturbances as rapidly as possible. The larger the loop bandwidth (the frequency of unity open-loop gain), the faster the controller responds to changes in the reference or disturbances in the loop.

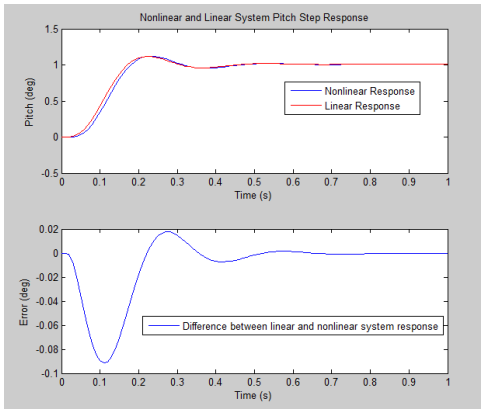


Figure 7 Comparison of Linear and Nonlinear models

- Adequate robustness — The loop design has enough gain margin and phase margin to allow for modeling errors or variations in system dynamics.[3]

We use a step or a Bode plot to verify the tuned controller. We update the controller gains in the model from the tuner's interface, tuning the roll and the pitch commands. The altitude is then tuned to achieve a hover control.

We validate the tuned response by comparing the linear step response, exported from the PID tuner tool (figure 6), to the nonlinear model step response simulated in the Simulink model. This analysis shows, through overlay and difference of the responses (figure 7), that the nonlinear and linear responses

compared well. These tests can be repeated as the nonlinear model is refinedusing data from the hardware to assure the control design linear models match the SimMechanics based nonlinear physical models.

The final step for this particular system is deploying the controller to the hardware, which in this case is a multiple step process:

1) Implement the controller in the full quadrotor model[1]

2) Determine additional control logic necessary for system calibration on startup

3) Generate and compile code for the controller model and deploy to the control unit on the vehicle

V. Next Steps

When the hardware is acquired we can more precisely determine the relationship between PWM inputs, electric motor torque, and thrust generation using system identification methods. Once this is completed the control system will be fine tuned and deployed to the vehicle. Flight testing the vehicle and comparing the results to the simulation will show how well the control design works. At this point the control can be expanded to achieve a number of tasks such as advanced tracking or maneuvers not currently possible with the basic control scheme.

VI. Conclusion

In this paper, we have shown that utilizing CAD data import capabilities to define the dynamics of a quadrotor system simplifies the overall design process by bypassing the process of deriving the differential equations for the system. After the modeling was completed, linear analysis tools were used to design a preliminary controller. The linear model compared well with the nonlinear model. This proves the linearization tool, as part of the PID tuner, was able to accurately linearize the different physical components within the model. As the fidelity of the model improves with the acquisition of hardware these same tools will continue to be applied to design the control system. Additional tools from this design environment can be employed for system identification, code generation, and deployment to the hardware[1].

Appendix

A. Sample of XML Code Created with SimMechanics Link software

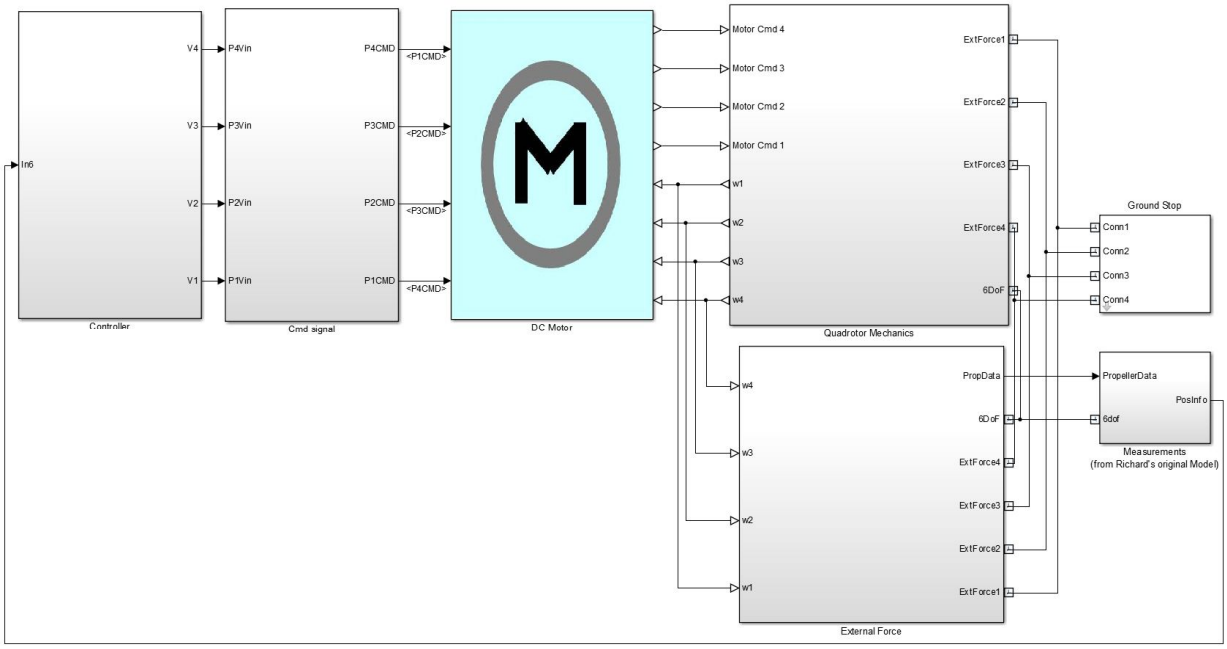
```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<SimMechanicsImportXML version="1.0" xmlns="urn:mathworks:simmechanics:import" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<Created by="" on="03/26/13||14:19:39" using="SimMechanics Link Version 4.2" from="SolidWorks 20.3.0"/>
<ModelUnits mass="kilogram" length="inch"/>
<DataUnits mass="kilogram" length="meter"/>
<RootAssembly name="QUADROTOR" uid="QUADROTOR" version="110">
<AssemblyFile name="QUADROTOR.sldasm" type="SolidWorks Assembly"/>
<InstanceTree>
<Instance name="BOARD-1" uid="BOARD-1" entityUid="BOARD*:*Default">
<Transform>
<Rotation>-0.707107 0.707107 0 -0.707107 -0.707107 0 0 0 1</Rotation>
<Translation>0 0 0.0070358</Translation>
</Transform>
</Instance>
<Instance name="ASSY, ROTOR END-1" uid="ASSY, ROTOR END-1" entityUid="ASSY, ROTOR END" rigid="true">
<Transform>
<Rotation>1 0 0 0 1 0 0 0 1</Rotation>
<Translation>0.0021844 -1.4e-006 1.4e-006</Translation>
</Transform>
<Instance name="TUBE END-1" uid="TUBE END-1" entityUid="TUBE END*:*Default">
<Transform>
```

```
<Rotation>0 -1 0 1 0 0 0 0 1</Rotation>
<Translation>0.083058 0 0</Translation>
</Transform>
<VisualProperties>
<Ambient r="0.25098" g="0.25098" b="0.25098" a="1"/>
<Diffuse r="0.25098" g="0.25098" b="0.25098" a="1"/>
<Specular r="0.25098" g="0.25098" b="0.25098" a="1"/>
<Emissive r="0" g="0" b="0" a="1"/>
<Shininess>0.31</Shininess>
</VisualProperties>
</Instance>
<Instance name="TUBE-1" uid="TUBE-1" entityUid="TUBE*:*Default">
<Transform>
<Rotation>1 0 0 0 1 0 0 0 1</Rotation>
<Translation>0 0 0</Translation>
</Transform>
<VisualProperties>
<Ambient r="0.25098" g="0.25098" b="0.25098" a="1"/>
<Diffuse r="0.25098" g="0.25098" b="0.25098" a="1"/>
<Specular r="0.25098" g="0.25098" b="0.25098" a="1"/>
<Emissive r="0" g="0" b="0" a="1"/>
<Shininess>0.31</Shininess>
</VisualProperties>
</Instance>
<Instance name="MOTOR-1" uid="MOTOR-1" entityUid="MOTOR*:*Default">
<Transform>
<Rotation>1 0 0 0 1 0 0 0 1</Rotation>
<Translation>0.0975336 0 0.0113538</Translation>
</Transform>
</Instance>
<Instance name="CARRIER-1" uid="CARRIER-1" entityUid="CARRIER*:*Default">
<Transform>
<Rotation>1 0 0 0 1 0 0 0 1</Rotation>
<Translation>0.108597 0 -0.0076962</Translation>
</Transform>
<VisualProperties>
<Ambient r="0.25098" g="0.25098" b="0.25098" a="1"/>
<Diffuse r="0.25098" g="0.25098" b="0.25098" a="1"/>
<Specular r="0.25098" g="0.25098" b="0.25098" a="1"/>
<Emissive r="0" g="0" b="0" a="1"/>
<Shininess>0.31</Shininess>
</VisualProperties>
</Instance>
<Instance name="GEAR-1" uid="GEAR-1" entityUid="GEAR*:*Default">
<Transform>

…

</VisualProperties>
</Part>
</Parts>
</SimMechanicsImportXML>
```

B. System Level Block Diagram of the Model



Acknowledgments

The authors would like to acknowledge the contribution of the team at Aero Analysis LLC for providing the CAD model and the hardware for our development efforts.

References

[1]Ruff, R., Stephens, C., Mahapatra, S., "Applying Model-Based Design to Large-Scale Systems Development: Modeling, Simulation, Test, & Deployment of a Multirotor Vehicle," Proceedings of AIAA 2012

[2]Goel, R., Shah, S., Gupta, N. and Ananthkrishnan, N., "Modeling, Simulation and Flight Testing of an Autonomous Quadrotor," *Proceedings of ICEAE 2009*

[3]MathWorks. Simulink® Control Design User's Guide. Natick : MathWorks, 2013.

[4]MathWorks. SimMechanics User's Guide. Natick : MathWorks, 2013.

[5]MathWorks. Simulink® User's Guide. Natick : MathWorks, 2013.

[6]Turevskiy, A, Gage, S and Buhr, C. *Model-Based Design of a New Light-weight Aircraft.* South Carolina : AIAA Modeling and Simulation Technologies Conference and Exhibit, 2007.