Johnathan Larymore

4/17/19

PHY 420

Research Report

This paper is to delve into the engineering challenges of developing a hover drone with the aid of a computer simulation model. This 2D Drone simulation was programmed using the python language and displayed via VPython. This 2D simulation, while impractical in reality, is important as it allows insight on the behavior of a real 3D drone.

The first place to start when making a computer model of a physical system is to first interpret the physical system and associate the appropriate physical equations of the model. The drone’s center of mass will be treated as a point mass which obeys Newtons Laws. This center of mass will also be treated as the center of a rigid body object which behaves according to the rotational representation of Newtons Laws.

Although incomplete, Newtons Laws yield over 99.9% accurate results for the position, velocity and acceleration of an object, with a known mass undergoing a net force. Knowing these net forces and applying Newtons 2nd Law, one can integrate with respect to time determine the change in velocity and again to determining the displacement. When given initial values for both velocity and position in addition of knowing what forces are present, one can determine an object’s coordinates at all times. This can be achieved by a variety of numerical time step integration techniques from an imported package or algorithms such as the Euler and Runge-Kutta, which are better suited as they require no additional imports and can be reproduced in any other code platform.

In constructing this model, Python was used as it is an interpretive language and easier understood. VPython works in conjunction with the base Python language and was used because of its ability to create and handle objects along with its visual capabilities. To iterate the motion of the drone, the Runge-Kutta algorithm was chosen. This time step method, while slower than the Euler method, provides a more precise solution than the Euler method, with minimal error due to the size of the time step. This is accomplished by preforming a weighted average of four mini steps for each time step. Within the Runge-Kutta method, Newtons laws will be applied to the system, and it will calculate the new the position, velocity, and acceleration along with the rotational values of the drone from their previous values.

How to set up code (more to come)

Now with a model code constructed, the first test can be performed to verify its accuracy to that of expected theoretical values of known systems. The first test was to verify that the drone acts the same as a free-falling object. This was done by applying only the gravitational force in the vertical direction. Within the wile loop the simulation was set to run and then stop after ten seconds. At this time, the drone’s position and velocity were recorded and compared to the corresponding expectational values which can be obtained using the Kinematic equations for contact acceleration. Test one successfully verified that the drone acts as an object in free fall when no other forces are applied.

The second test was to ensure that the drone behaved as a rigid body. This was done by first temporarily zeroing out the accelerations in both the vertical and horizontal directions to ensure only rotational motion would occur. A force of 1N from a thruster was applied to one side of the drone resulting in a net torque of 5NM which caused the drone to rotate. Similar to test one, the motion of a rigid body with constant angular acceleration was determined using the rotational version of the kinematic equations. Test two successfully confirmed that the drone behaved as a rigid body with the same net torque and moment of inertia.

Now that the virtual drone appears to obey Newtons laws of motion for a point mass and a rigid body, further exploration became possible.

Critique by inserting two forces (thrusters) on the sides of 2D drone equal to w/2 (to make it hover without rotation). Pass

Test unbalanced drone in both directions. Pass

Assign thruster force values both to a slider widget on menu.

Test. Worked but was very hard to control.

Critique (reduce thrusters to have a smaller range around w/2 (hover) to reduce the torque and make rotation less sensitive)

Test. Improved control but still not practical